



UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Problemas localmente verificables parametrizados por treewidth, clique-width y mim-width

Tesis presentada para optar al título de
Doctora de la Universidad de Buenos Aires
en el área Ciencias de la Computación

Carolina Lucía Gonzalez

Directora: Flavia Bonomo

Consejero de estudios: Javier Marengo

Buenos Aires, 15 de abril de 2024

Problemas localmente verificables parametrizados por treewidth, clique-width y mim-width

Intuitivamente, un problema localmente verificable es un problema de partición de vértices (o, equivalentemente, de coloreo de vértices) para el cual una solución puede ser verificada simplemente chequeando una determinada propiedad local para cada vértice, es decir, una propiedad que involucra solamente la solución restringida al vértice y a sus vecinos. Este es el caso de diversas variantes de los problemas de dominación, conjunto independiente y k -coloreo, entre otros. Existen numerosos marcos generales que incluyen un gran subconjunto de estos problemas, para los cuales se propusieron algoritmos para resolverlos eficientemente en varias clases de grafos.

En esta tesis definimos un nuevo marco para problemas localmente verificables, formalizando la noción intuitiva que tenemos de ellos para luego analizar bajo qué circunstancias podemos proponer algoritmos eficientes que los resuelvan. Los algoritmos propuestos se basan en programación dinámica y para ellos calculamos sus complejidades parametrizadas por treewidth, clique-width y mim-width. Los resultados obtenidos generalizan aquellos de marcos definidos anteriormente. Mostramos además cómo modelar dentro de nuestro marco diversos problemas de la literatura, como ser $[k]$ -dominación romana, k -comunidad y dominación Grundy, probando así que son FPT o XP parametrizados por treewidth, clique-width o mim-width. Más aún, proponemos esquemas de aproximación polinomiales para algunos problemas localmente verificables en grafos planares.

Palabras claves: problema localmente verificable, treewidth, clique-width, mim-width, complejidad parametrizada, esquema de aproximación polinomial.

Locally checkable problems parameterized by treewidth, clique-width and mim-width

Intuitively, a locally checkable problem is a vertex partition problem (or, equivalently, a vertex coloring problem) for which a solution can be verified simply by checking a certain local property for each vertex, that is, a property that involves only the solution restricted to the vertex and its neighbors. This is the case of several variants of domination, independent set and k -coloring, among other problems. There exist numerous frameworks that include a large subset of these problems, for which algorithms have been proposed to solve them efficiently in various graph classes.

In this thesis we define a new framework for locally checkable problems, formalizing the intuitive notion we have of them to then analyze under what circumstances we can propose efficient algorithms to solve them. The proposed algorithms are based on dynamic programming and we computed their complexities when parameterized by treewidth, clique-width and mim-width. The results obtained generalize those of previously defined frameworks. We further show how to model within our framework several problems in the literature, such as $[k]$ -Roman domination, k -community and Grundy domination, thus proving that they are FPT or XP when parameterized by treewidth, clique-width or mim-width. Moreover, we propose polynomial-time approximation schemes for some locally checkable problems in planar graphs.

Keywords: locally checkable problem, treewidth, clique-width, mim-width, parameterized complexity, polynomial-time approximation scheme.

Índice general

Resumen	I
Abstract	III
Introducción	IX
1. Preliminares	1
1.1. Complejidad computacional	1
1.1.1. Complejidad parametrizada	2
1.2. Autómatas determinísticos de estados finitos	3
1.3. Nociones algebraicas básicas	4
1.4. Conjuntos de pesos	5
1.5. Definiciones básicas de teoría de grafos	6
1.6. Parámetros de anchura en grafos	11
1.6.1. Treewidth	11
1.6.2. Clique-width	15
1.6.3. Árboles binarios de descomposición: carvingwidth, mim-width y otros parámetros	16
1.6.4. Relaciones entre los parámetros	18
2. Problemas localmente verificables	23
2.1. Definición	23
2.2. Ejemplos	24
2.3. Propiedades globales	26
2.3.1. Tamaño	26
2.3.2. Conectividad	26
2.3.3. Aciclicidad	27
2.3.4. Composición de condiciones locales	27
2.4. Restricciones consideradas	28
2.4.1. Sistema de vecindades parciales	28
2.4.2. Conteo de colores	30
2.4.3. d -estabilidad	32
2.4.4. m -modularidad	33
3. Parametrización por treewidth	35
3.1. Algoritmo principal	35
3.1.1. Notación y definiciones	35
3.1.2. Función a computar	37

3.1.3.	Análisis de complejidad	41
3.2.	Propiedades globales	42
3.2.1.	Tamaño	42
3.2.2.	Conectividad positiva	43
3.2.3.	Aciclicidad positiva	45
3.2.4.	Composición de condiciones locales	45
3.3.	El caso $r > 1$	47
3.3.1.	Parametrización por treewidth más grado máximo, o por carvingwidth	47
3.3.2.	Transformación a problemas 1-localmente verificables	48
3.4.	Límites	49
4.	Parametrización por clique-width	53
4.1.	Algoritmo	53
4.2.	Propiedades globales	61
4.2.1.	Composición de condiciones locales	61
4.3.	El caso $r > 1$	67
4.4.	Límites	67
5.	Parametrización por mim-width	71
5.1.	Algoritmo	71
5.1.1.	Extensión a distancias $r > 1$	79
5.1.2.	Extensión a propiedades globales de composición de condiciones locales	80
5.2.	Límites	81
6.	Esquemas de aproximación de tiempo polinomial (PTAS)	85
6.1.	Preliminares	86
6.2.	Restricciones consideradas	88
6.3.	Problemas de minimización	89
6.4.	Problemas de maximización	91
7.	Aplicaciones	95
7.1.	Dominación romana	95
7.2.	Dominación dual	98
7.3.	Coloreos felices	100
7.4.	Coloreo localmente acotado	101
7.5.	Coloreos libres de conflictos	101
7.6.	Empaquetamientos generalizados	102
7.7.	Problemas de partición en subgrafos densos	103
7.7.1.	Comunidades	103
7.7.2.	PDS	105
7.7.3.	Quasi-cliques	105
7.8.	b-coloreo	107
7.9.	Coloreo aditivo	108
7.10.	Mínima violación cromática	108
7.11.	Dominación Grundy	109
7.12.	Problemas que involucran aristas	111

8. Relación con otros marcos generales	113
8.1. Problemas LCVS y LCVP	113
8.2. Problemas LCC y ECC	114
8.3. Lógica A&C DN	119
9. Conclusiones y trabajo futuro	127
A. Definiciones de problemas	131
B. Abreviaturas	135
B.1. Complejidad	135
B.2. Parámetros de anchura	135
B.3. Marcos de problemas	136
B.4. Otros	136
C. Resumen de nuestro marco	137
C.1. Elementos	137
C.1.1. Propiedades globales	137
C.2. Restricciones	137
Bibliografía	139

Introducción

El origen de la teoría de grafos se remonta a 1736, cuando Leonhard Euler propuso una solución al famoso problema de los puentes de Königsberg, el cual consistía en determinar si era posible atravesar todos los puentes de la ciudad exactamente una vez y volver al punto de inicio. Si bien Euler no definió formalmente esta rama de la matemática, en su artículo enunció lo que es considerado como el primer teorema de la teoría de grafos. No fue sino hasta 140 años más tarde que se introdujo el término *grafo* [162]. En 1936 König publicó el primer libro enteramente dedicado a la teoría de grafos [127], precedido por la monografía de Sainte-Laguë de 1926 (considerada el “0-ésimo” libro de la teoría de grafos) [100].

Un grafo no es más que un conjunto de elementos llamados *vértices* y un conjunto de pares de vértices (*aristas*), que usualmente ilustramos utilizando puntos conectados por líneas. Esta simple definición es útil para modelar un sinnúmero de situaciones. Por ejemplo, podemos pensar en ciudades conectadas por rutas, usuarios de una red social unidos por su relación de amistad, antenas y sus interferencias, laberintos, juegos (como el sudoku), y la lista podría seguir.

Veamos un ejemplo concreto: supongamos que queremos colocar la menor cantidad posible de caballos en un tablero de ajedrez de modo que todas las casillas sean amenazadas. Podemos definir un vértice por cada casilla del tablero y una arista entre dos vértices si al ubicar un caballo en una de las correspondientes casillas entonces la otra casilla resulta amenazada, como podemos ver en la Figura 1. Luego, el problema consiste en encontrar un subconjunto mínimo de vértices tales que todos los demás vértices estén conectados a al menos uno de ellos. Trasladado a cualquier grafo, este problema se conoce como el de *mínimo conjunto dominante* y es uno de los más antiguos y estudiados de la teoría de grafos. Numerosas aplicaciones se modelan de forma similar pero con restricciones adicionales, definiendo así *variantes* del problema de dominación (tan solo en [108] podemos encontrar más de 80 de estas variantes). Otros problemas relevantes en teoría de grafos que podemos mencionar son los de *k-coloreo* y *máximo conjunto independiente*. Dado un entero k , en el problema de k -coloreo queremos determinar si existe una asignación de colores a los vértices, utilizando a lo sumo k colores, de modo que si dos vértices están conectados por una arista entonces reciban colores distintos (ver Figura 2a). En cambio, para el problema de máximo conjunto independiente, queremos determinar la máxima cantidad de vértices que podemos escoger de manera que ningún par de dichos vértices esté conectado por una arista (ver Figura 2b).

Aunque todos los problemas que mencionamos puedan parecer muy diferentes, tienen dos características fundamentales en común. En primer lugar, son problemas de *partición de vértices*, es decir, problemas en los que debemos clasificar a los vértices en distintas categorías (como ser parte o no del conjunto dominante o independiente, o los

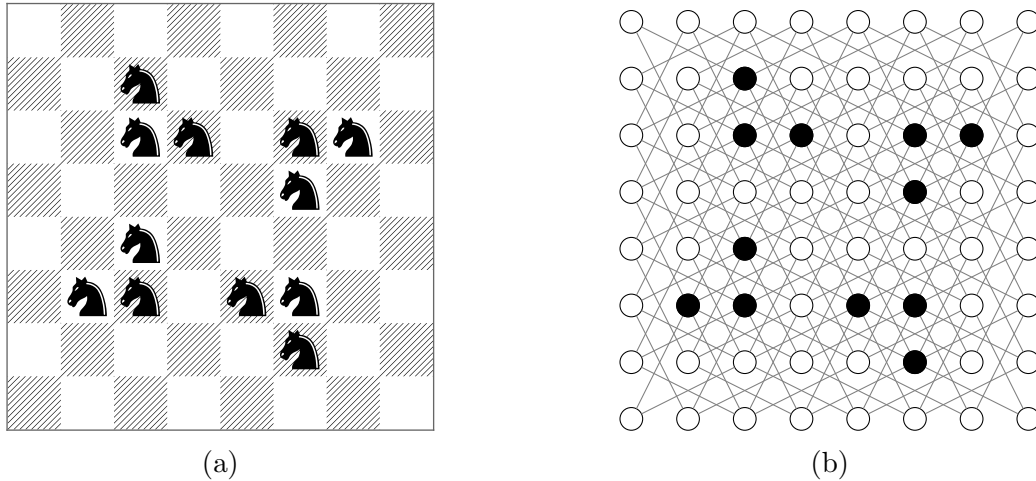


Figura 1: El problema de ubicar caballos en un tablero de ajedrez de manera de amenazar todas las casillas. A la izquierda, el tablero con las piezas ubicadas. A la derecha, la representación como un grafo.

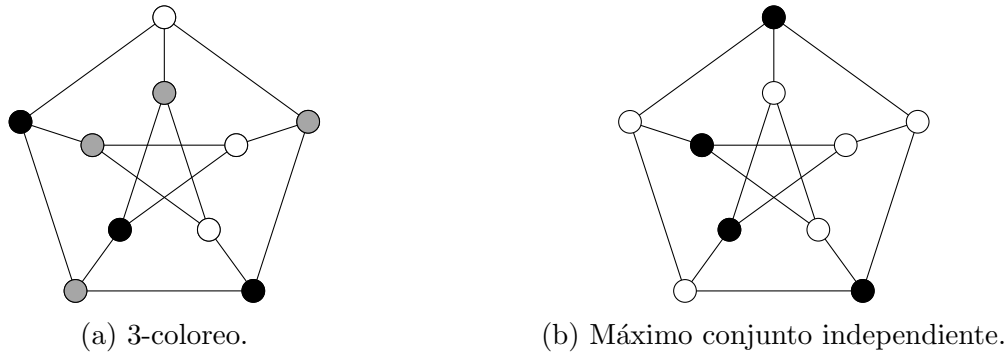


Figura 2: Ejemplos de coloreo y conjunto independiente en un grafo.

colores asignados). Además, existe una *propiedad local* que se debe satisfacer para cada vértice, esto es, una propiedad que involucra sólo al vértice y a aquellos conectados a este vértice (sus *vecinos*). Por ejemplo, para el problema de conjunto dominante hay que verificar, para cada vértice v , si v o alguno de sus vecinos pertenece al conjunto dominante, mientras que para el problema de coloreo hay que verificar si v tiene un color distinto al de sus vecinos. Los problemas con estas características se conocen como *problemas localmente verificables*. Esta tesis está abocada precisamente a ellos. Mínimo conjunto dominante, k -coloreo y máximo conjunto independiente son sólo algunos ejemplos de problemas localmente verificables, pero también podemos mencionar, por ejemplo, problemas de *empaquetamiento* (donde se quiere asignar valores a los vértices de modo que para cada vértice la suma de los valores en su vecindad no exceda un determinado umbral). Dadas sus múltiples aplicaciones en la vida real, resulta de interés desarrollar algoritmos que puedan resolver cada instancia de estos problemas en un tiempo razonable.

La *complejidad temporal* de un algoritmo se mide según la cantidad de pasos que efectúa y generalmente se expresa en función del tamaño de la entrada (por ejemplo, si la entrada es un grafo podríamos considerar al tamaño de la entrada como la cantidad de vértices y aristas). Un algoritmo es *de tiempo polinomial* si su complejidad temporal

tiene como cota superior un polinomio $p(n)$, siendo n el tamaño de la entrada. Si este polinomio es de grado 1, decimos que el algoritmo es *de tiempo lineal*.

Llamamos *problemas de decisión* a aquellos cuya respuesta es “sí” o “no”, en contraposición a los *problemas de optimización*, donde se busca el valor óptimo que satisfaga una determinada condición. A todo problema de optimización se le puede asociar un problema de decisión: se incluye un valor k en la entrada y se pregunta si existe una solución de valor al menos tan bueno como k . La *clase de problemas P* contiene a los problemas de decisión que se pueden resolver mediante algoritmos de tiempo polinomial, problemas a los que también llamaremos *tratables*. La clase *NP* es aquella que contiene a los problemas de decisión para los cuales, para toda instancia en que la respuesta es “sí”, existe un *certificado* de la misma y un algoritmo de tiempo polinomial que lo verifica. ¿Es posible resolver cualquier problema de esta clase en tiempo polinomial? Dicho de otra manera, ¿ $P = NP$? Esta pregunta es una de las más importantes y difíciles en las ciencias de la computación y aún no ha sido respondida, aunque en general se cree que la respuesta sería no.

Los problemas *NP-completos* (abreviado *NP-c*) conforman un subconjunto especial de problemas NP, con la característica de que si alguno de ellos puede ser resuelto por un algoritmo de tiempo polinomial (es decir, si pertenece a P) entonces todos los problemas en NP también pertenecen a P. Por otro lado, los problemas *NP-difíciles* (abreviado *NP-h*) son problemas que no necesariamente pertenecen a NP y que son “al menos tan difíciles” como los NP, en el sentido de que si un problema NP-h se puede resolver en tiempo polinomial entonces cada uno de los problemas NP también. Luego, cuando nos encontramos con un problema NP-completo o NP-difícil, no es esperable obtener un algoritmo polinomial que lo resuelva (ya que si lo hiciéramos entonces estaríamos respondiendo a la pregunta de $P = NP$).

No obstante, los problemas NP-difíciles se pueden abordar desde una perspectiva diferente. Podemos, por ejemplo, relajar el requisito de la optimalidad de la solución, dando lugar al desarrollo de *algoritmos aproximados*. Otra posibilidad es considerar un subconjunto concreto de las instancias. En el caso de los problemas de grafos, esto implica resolver el problema en una determinada *clase de grafos* (un conjunto infinito de grafos que comparten alguna característica). En esta tesis aplicaremos ambos enfoques y además un tercero que ha cobrado relevancia recientemente: la *complejidad parametrizada*. En este último caso, la complejidad de los algoritmos se expresa en términos de dos valores: el tamaño de la entrada (como es usual) y otro valor más, llamado *parámetro*. El objetivo es determinar cómo este parámetro afecta la tratabilidad del problema. Cuando obtenemos una complejidad de la forma $f(k)n^c$, siendo n el tamaño de la entrada, k el valor del parámetro, f alguna función y c alguna constante, decimos que el problema es *FPT parametrizado por k*. Por otro lado, si la complejidad es $n^{f(k)}$, decimos que es *XP parametrizado por k*. Notemos que, en ambas situaciones, el problema es resoluble en tiempo polinomial para cada valor fijo de k . Podemos encontrar casos en los que el problema sigue siendo NP-difícil incluso para valores constantes de k y entonces decimos que es *para-NP-difícil*. Discutiremos este tema con mayor profundidad en la Sección 1.6.

Los problemas de decisión asociados a dominación, k -coloreo y conjunto independiente son NP-completos [92]. A pesar de ello, se conocen algoritmos de tiempo polinomial para resolver estos problemas en diversas clases de grafos, entre las que podemos mencionar árboles [60, 94], cografos [96, 108] y grafos de intervalo [42, 94], aunque en

algunas clases los tres problemas siguen siendo NP-completos (por ejemplo, los grafos planares [92]), y en otros algunos problemas son tratables y los otros no (por ejemplo, en grafos cordales el problema sobre dominación es NP-completo [42] pero tanto coloreo como conjunto independiente son resolubles en tiempo polinomial [94]).

Ciertas clases de grafos se definen en función del valor de un *parámetro de anchura*. El más conocido de estos parámetros es quizás la *anchura de árbol (treewidth)*, definida en 1983 por Robertson y Seymour [152], y previamente considerada bajo diferentes nombres por Bertelè y Brioschi [29] y Halin [106]. Intuitivamente, este parámetro indica qué tan lejos está un grafo de ser un árbol o bosque. Algunas clases de grafos de treewidth acotada incluyen: bosques (treewidth 1), pseudo-bosques, cactus, grafos outerplanar y grafos serie-paralelo (treewidth a lo sumo 2), grafos Halin y redes Apollonian (treewidth a lo sumo 3) [34, 45, 126, 151], e incluso los grafos de flujo de control que surgen de la compilación de programas estructurados también tienen treewidth acotada [130, 167]. Para clases de grafos donde este parámetro es una constante, los problemas de dominación, k -coloreo y conjunto independiente son resolubles en tiempo lineal [13]. Más específicamente, estos problemas son FPT parametrizados por treewidth. Otros parámetros de anchura relevantes son *clique-width* y *mim-width*. Estos tres parámetros tienen árboles asociados que nos permiten recorrer el grafo de manera ordenada y, fundamentalmente, facilitan la aplicación de la *programación dinámica* (una técnica algorítmica que se basa en combinar sub-soluciones recursivamente para obtener una solución).

Es importante destacar que muchos de los algoritmos para resolver problemas localmente verificables, en ciertas clases de grafos, guardan gran similitud (por ejemplo, los algoritmos de [13] son esencialmente los mismos para tres de los problemas allí planteados, o los algoritmos de [134, 176] para resolver algunas variantes del problema de dominación en árboles también son similares entre sí). Por este motivo, surge naturalmente la pregunta de si podemos definir un *marco general (framework)* que incluya un gran subconjunto de estos problemas y proponer un único algoritmo que nos permita resolver eficientemente cualquiera de esos problemas en algunas clases de grafos, evitando así desarrollar independientemente un algoritmo específico para cada problema. Con este espíritu, en [32] Bodlaender definió los problemas de *composición de condiciones locales (local condition composition)* y *composición de condiciones de aristas (edge condition composition)*, abreviados LCC y ECC respectivamente, por sus siglas en inglés. Bodlaender mostró, además, algoritmos polinomiales para resolver problemas LCC en clases de grafos de treewidth y grado acotados, y problemas ECC en clases de grafos de treewidth acotada. En [164] Telle definió los problemas *localmente verificables de partición de vértices (locally checkable vertex partitioning, LCVP)* y probó que son FPT parametrizados por treewidth. Más adelante, en [50], Bui-Xuan, Telle y Vatshelle presentaron un algoritmo de programación dinámica para resolver problemas LCVP que, junto con los resultados de [23], prueba que estos problemas son XP parametrizados por mim-width (dado un árbol asociado). En [53], Cattaneo y Perdrix definieron una variante de los problemas LCVP que nos permite lidiar con algunas propiedades que no son necesariamente localmente verificables (por ejemplo, que una clase de la partición sea conexa). Los célebres teoremas de Courcelle [67, 68] establecen que si un problema de grafos se puede expresar en lógica monádica de segundo orden MSO_2 entonces es FPT parametrizado por treewidth, mientras que si es expresable en MSO_1 entonces es FPT parametrizado por clique-width. Un resultado

similar fue obtenido recientemente por Bergougnoux, Dreier y Jaffke en [26], quienes presentaron una lógica derivada de MSO_1 , llamada *A&C DN*, y probaron que todo problema expresable en dicha lógica es XP parametrizado por *mim-width* (dado un árbol asociado).

En esta tesis se presenta un nuevo marco para problemas localmente verificables pero con un enfoque diferente. El objetivo fue formalizar la definición de problemas localmente verificables para luego poder analizar bajo qué circunstancias se pueden proponer algoritmos eficientes para resolverlos. Los algoritmos propuestos se basan en programación dinámica y para ellos se calculan sus complejidades parametrizadas por *treewidth*, *clique-width* y *mim-width*. Además, para cada parámetro, se muestran ejemplos de problemas para-NP-difíciles que no cumplen con las restricciones impuestas para dicho parámetro. También se considera un conjunto específico de *propiedades globales* (incluyendo, por ejemplo, que cierta clase de la partición induzca un grafo conexo o acíclico) y propiedades locales a distancias mayores a 1. Se demuestra que los resultados obtenidos generalizan aquellos de [32, 50, 53], y se estudia con mayor profundidad su relación con los resultados de [26]. Se muestra, además, cómo modelar numerosos problemas de la literatura dentro del nuevo marco, como ser dominación Grundy, k -comunidad y $[k]$ -dominación romana, entre otros. Como resultado inmediato, se obtienen los primeros algoritmos de tiempo polinomial para resolver estos problemas en grafos de *treewidth* acotada, *clique-width* acotada o *mim-width* acotada (según el caso). Más aún, se proponen *esquemas de aproximación polinomiales* para un gran subconjunto de problemas localmente verificables en grafos planares.

Organización de la tesis

La tesis está organizada de la siguiente manera. El **Capítulo 1** contiene las definiciones y los resultados preliminares necesarios para nuestro trabajo, además de dos nuevos resultados sobre parámetros de anchura y grado. La noción central de esta tesis (el nuevo marco para los problemas localmente verificables) es formalmente presentada en el **Capítulo 2**. En el **Capítulo 3**, proponemos un algoritmo parametrizado por *treewidth* para resolver problemas localmente verificables y analizamos su complejidad, también incluimos algunas propiedades globales y distancias mayores a 1. Realizamos un estudio similar para los parámetros de *clique-width* en el **Capítulo 4**, y *mim-width* en el **Capítulo 5**. En el **Capítulo 6**, estudiamos la técnica de Baker aplicada a problemas localmente verificables, obteniendo así esquemas de aproximación polinomiales (PTAS) en grafos planares para un gran subconjunto de estos problemas. En el **Capítulo 7** presentamos una serie de aplicaciones de los resultados, es decir, mostramos cómo modelar problemas previamente definidos en la literatura utilizando nuestro marco y analizamos las características de dicho modelo para determinar si satisface las restricciones para que el problema sea FTP o XP parametrizado por *treewidth*, *clique-width* o *mim-width*, y para que admita un PTAS en grafos planares. El **Capítulo 8** está dedicado a estudiar la relación que existe entre el nuevo marco que definimos para problemas localmente verificables y otros marcos previamente definidos (problemas LCVP, LCC y ECC, y lógica A&C DN). Concluimos con una discusión sobre los resultados obtenidos y las posibles líneas de trabajo a futuro. Los **Apéndices A, B y C** contienen, respectivamente, una lista de definiciones de problemas mencionados a lo largo de esta tesis, las principales abreviaturas y un resumen del marco presentado.

Publicaciones

Esta tesis está fuertemente basada en los artículos [16, 41, 103], los cuales fueron desarrollados en el transcurso de mi doctorado. El primero de ellos ([41]) es en coautoría con mi directora y los restantes ([16, 103]) son resultado de mis visitas a la Universidad de Fribourg, Suiza. No obstante, se incluyen varias modificaciones y extensiones, como se detalla a continuación. Salvo expresamente indicado, estas modificaciones son de mi exclusiva autoría.

- En el Capítulo 1 se incluye un nuevo resultado (Teorema 1.6.22) que relaciona el parámetro carvingwidth con el parámetro sim-width y el grado máximo de un grafo. Este resultado fue obtenido en colaboración con Felix Mann.
- En el Capítulo 2, y a lo largo de toda la tesis, se unifican las definiciones de los marcos presentados en los artículos [16, 41, 103]. Además, se presenta una nueva restricción que se utilizará en el Capítulo 4.
- El Capítulo 3 incluye el algoritmo presentado en [41] con una modificación: en lugar de considerar pesos $w_{v,i}$ para cada vértice v y color i , ahora existe una función de peso w que toma un vértice y un coloreo de la vecindad del vértice (similar a la función *check*). También se adapta la notación de las extensiones con propiedades globales a la del artículo [16], se extiende la sección sobre el caso $r > 1$, y se agregan más ejemplos de problemas difíciles que no cumplen con las restricciones presentadas.
- En el Capítulo 4 se extiende el algoritmo presentado en [16] a una nueva restricción (la existencia de un sistema de vecindades parciales universal), se generaliza la propiedad global de tamaño a la propiedad global de composición de condiciones locales, se analiza el caso $r > 1$, y se incluyen ejemplos de problemas difíciles que no cumplen con las restricciones.
- El Capítulo 5 presenta el algoritmo dado en [103] junto a su extensión a mayores distancias sin modificaciones sustanciales, pero se agrega una extensión a propiedades globales de composición de condiciones locales. Los ejemplos de problemas para-NP-difíciles parametrizados por mim-width dados en [103] también se presentan aquí sin modificaciones.
- Los resultados del Capítulo 6 son nuevos y surgieron gracias a que el Prof. Min Chih Lin me sugirió estudiar esquemas de aproximación polinomiales para problemas localmente verificables.
- El Capítulo 7 mayormente combina los ejemplos presentados en [16, 41, 103] pero también incluye algunos nuevos.
- Las relaciones con otros marcos ya estudiadas en [16, 41, 103] se presentan sin mayores modificaciones en el Capítulo 8, sólo se incluyen algunos comentarios adicionales.

Capítulo 1

Preliminares

1.1. Complejidad computacional

Un *problema* π consiste en un conjunto de instancias D_π , que contiene todas las posibles entradas para el problema, y una pregunta sobre esas instancias. Decimos que un algoritmo resuelve un problema π si ese algoritmo puede ser aplicado a cualquier instancia $I \in D_\pi$ y se garantiza que siempre producirá la respuesta correcta para dicha instancia I . Un *problema de decisión* es aquel cuyas respuestas son “sí” o “no”. Un problema es de *optimización* cuando la respuesta es una solución óptima para el problema planteado. En general, un problema de decisión puede ser derivado de uno de optimización. Por ejemplo, si el problema de optimización pregunta por una estructura de cierto tipo que minimice cierto costo (como puede ser un camino de longitud mínima), podemos asociarlo con el problema de decisión que incluya adicionalmente un número c como parámetro y pregunte si existe una estructura de ese tipo que tenga costo menor o igual a c (en el ejemplo, un camino de longitud menor o igual a c).

Los problemas también se agrupan según cuán costoso es resolverlos. La *función de complejidad temporal* de un algoritmo expresa sus requerimientos temporales. Intuitivamente, podemos verla como la cantidad de pasos que efectúa hasta obtener la respuesta. Habitualmente es expresada como una función en términos de una sola variable: el tamaño de una instancia del problema. Para medir el tamaño de una instancia, observemos que la descripción de la misma puede ser vista como una cadena finita de símbolos elegidos de un alfabeto finito de entrada. Aunque hay varias formas distintas en que las instancias de un problema dado pueden ser descriptas, vamos a suponer que elegimos por adelantado una en particular. Así, el tamaño de una instancia I de un problema π se define como el número de símbolos en la descripción de I .

Dada una función $f: \mathbb{N} \rightarrow \mathbb{N}$, el conjunto $O(f)$ se define como el conjunto de todas las funciones $g: \mathbb{N} \rightarrow \mathbb{N}$ para las cuales existen dos constantes n_0, c tales que $g(n) \leq c \cdot f(n)$ para todo $n > n_0$. Si $g \in O(f)$, es usual decir que “ g es $O(f)$ ”, o “ $g(n) \in O(f(n))$ ”, o “ $g(n)$ es $O(f(n))$ ”. Estos términos serán utilizados indistintamente a lo largo de este trabajo. Por momentos haremos abuso de notación y escribiremos $n^{O(x)}$ para indicar “ $O(n^{cx})$ para alguna constante c ”.

Un algoritmo *de tiempo polinomial* (o simplemente *polinomial*) se define como aquel que tiene una función de complejidad temporal que es $O(p(n))$ para algún polinomio p , donde n denota el tamaño de la instancia. Una *transformación (o reducción) polinomial* de un problema de decisión π_1 a un problema de decisión π_2 es una función

$f : D_{\pi_1} \rightarrow D_{\pi_2}$ que satisface las siguientes dos condiciones:

1. f es computable por un algoritmo de tiempo polinomial; y
2. para toda instancia $I \in D_{\pi_1}$, la respuesta a I en π_1 es “sí” si y sólo si la respuesta a $f(I)$ en π_2 es “sí”.

Los problemas de decisión que tienen algoritmos de tiempo polinomial que los resuelven se agrupan en la *clase P*. A estos problemas también los llamaremos *tratables* o *fáciles*, en contraposición con los demás problemas, que serán llamados *intratables* o *difíciles*. La *clase NP* contiene a los problemas de decisión que pueden ser resueltos polinomialmente por una máquina de Turing no determinística. Para no entrar en detalles que van más allá de esta tesis, podemos decir que la clase NP es aquella que contiene a los problemas de decisión para los cuales, para toda instancia en que la respuesta es “sí”, existe un *certificado* de la misma y un algoritmo polinomial que lo verifica. Por ejemplo, si el problema es determinar si existe una ruta de longitud menor o igual a 10km en un mapa, entonces un certificado es una ruta posible y la verificación es comparar su longitud con 10km. Si esta verificación se puede hacer en tiempo polinomial entonces el problema pertenece a NP.

En 1971, Cook [63] probó que un problema particular de la clase NP, el llamado *problema de satisfacibilidad booleana (SAT)*, tiene la propiedad de que todo problema en NP puede ser reducido polinomialmente a él. Esto significa que si SAT pudiera ser resuelto con un algoritmo polinomial entonces todos los otros problemas en NP también, y si algún problema en NP fuera intratable entonces también lo sería SAT. Cook sugirió que otros problemas en NP debían compartir con SAT esta propiedad de ser “los más difíciles” de la clase. Karp [125] y luego Garey y Johnson [92] presentaron más problemas como este y se definió entonces la clase de problemas *NP-completos* (abreviado *NP-c*). Formalmente, un problema π es NP-completo si π pertenece a NP y, para todo otro problema π' en NP, existe una transformación polinomial de π' a π . Si omitimos la primera condición, decimos que π es *NP-difícil* (o *NP-hard*, abreviado *NP-h*). Una clásica pregunta sin respuesta aún es si las clases P y NP son iguales. En otras palabras, ¿es posible resolver cualquier problema en la clase NP en tiempo polinomial?

1.1.1. Complejidad parametrizada

En un *problema parametrizado* la entrada se considera en dos partes: (x, k) , donde x es la *entrada principal* y k es un entero llamado *parámetro*. La función de complejidad de este tipo de problemas es expresada en términos del tamaño n de la entrada principal x y del parámetro k , de la forma $t(n, k)$. Según el tipo de función de complejidad, distinguimos a los problemas en clases, como las definidas a continuación.

Definición 1.1.1 (FPT). Un problema parametrizado es *tratable de parámetro fijo* (en inglés, *fixed-parameter tractable*, abreviado *FPT*) si existen: un algoritmo $\mathcal{A}$ que resuelve el problema, una función computable $f : \mathbb{N} \rightarrow \mathbb{N}$ y una constante $c \in \mathbb{N}$, tales que la función de complejidad temporal $t(n, k)$ del algoritmo $\mathcal{A}$ pertenece a $O(f(k)n^c)$. Cuando esta constante c es igual a 1, decimos que el problema es *FPT lineal*.

Definición 1.1.2 (XP). Un problema parametrizado es *polinomial en rebanadas* (en inglés, *slice-wise polynomial*, notado *XP*) si existen: un algoritmo $\mathcal{A}$ que resuelve el

problema y dos funciones computables $f, g: \mathbb{N} \rightarrow \mathbb{N}$, tales que la función de complejidad temporal $t(n, k)$ del algoritmo $\mathcal{A}$ pertenece a $O(f(k)n^{g(k)})$.

En caso de necesitar enfatizar cuál es el parámetro k considerado, diremos que un problema es FPT o XP *parametrizado por k* . Notemos que todo problema FPT parametrizado por k es también XP parametrizado por k (ya que $g(k)$ puede ser una función constante), y por [73, Observación 15.2], esta contención es estricta (es decir, que existen problemas XP que no son FPT). Por otro lado, notemos que si un algoritmo es de tiempo $O(f(k)n^{g(k)})$ entonces cada vez que fijamos k a una determinada constante, el algoritmo resulta de tiempo polinomial en n (de allí el nombre “polinomial en rebanadas”). En particular, si g es una función constante (digamos, $g(k) = c$) entonces, para cada valor fijo de k , la complejidad en n es la misma ($O(n^c)$). Esto no ocurre para otras funciones g . Por ejemplo, consideremos un algoritmo de tiempo $O(n^k)$. Cuando $k = 1$, este algoritmo es de tiempo lineal, pero cuando $k = 2$, este algoritmo es de tiempo cuadrático. Resulta entonces evidente que es preferible obtener algoritmos FPT por sobre XP.

De manera similar a las reducciones polinomiales, una *reducción FPT* de un problema parametrizado a otro transforma una instancia (x, k) del primer problema en una instancia (x', k') del segundo en tiempo $f(k)|x|^{O(1)}$, donde k' se calcula en función de k solamente. La clase de problemas parametrizados FPT es cerrada bajo este tipo de reducciones. Otras clases de problemas parametrizados que son cerradas bajo reducciones FPT son las llamadas $W[i]$, con i entero positivo. No daremos su definición formal dado que escapa de los objetivos de esta tesis, pero sí utilizaremos la siguiente propiedad [73]:

$$\text{FPT} \subseteq W[1] \subseteq W[2] \subseteq \dots \subseteq \text{XP}$$

Es una pregunta abierta si $W[1] = \text{FPT}$ y se considera que es incluso más difícil de responder que la pregunta de $P = NP$ [43]. Distinguimos a los problemas llamados *$W[1]$ -difíciles* (en inglés, *$W[1]$ -hard*, abreviado *$W[1]$ -h*), que son aquellos tales que todo problema en $W[1]$ puede reducirse de manera FPT a ellos. Dicho de otra manera, estos son problemas que “muy probablemente” no pertenezcan a FPT.

Cuando un problema parametrizado es NP-difícil incluso para un determinado valor constante del parámetro k , decimos que el problema es *para-NP-difícil* (en inglés, *para-NP-hard*, abreviado *para-NP-h*). Claramente, si $P \neq NP$, estos problemas no son XP.

1.2. Autómatas determinísticos de estados finitos

Presentaremos brevemente este tema, ya que sólo se utiliza como una herramienta en una de las demostraciones. Para más información sobre teoría de autómatas, ver [115].

Un *autómata determinístico de estados finitos* es una quintupla $(Q, \Sigma, \delta, q_0, F)$ que consiste de:

- Q : un conjunto finito de *estados*,
- Σ : un conjunto finito de *símbolos de entrada* (también llamado *alfabeto*),
- $\delta: Q \times \Sigma \rightarrow Q$: una *función de transición*,
- $q_0 \in Q$: un *estado inicial*, y

- $F \subseteq Q$: un conjunto de *estados finales* o *de aceptación*.

Decimos que un autómata $M = (Q, \Sigma, \delta, q_0, F)$ *acepta* una cadena $s_1 \dots s_n$, con $n \geq 1$, si y sólo si $s_i \in \Sigma$ para todo $i \in \{1, \dots, n\}$ y $\delta(\dots \delta(q_0, s_1) \dots, s_n) \in F$. Por ejemplo, el autómata $M_i = (\{q_0, q_1\}, \{1\}, \delta, q_0, \{q_1\})$, donde $\delta(q_0, 1) = q_1$ y $\delta(q_1, 1) = q_0$, es el autómata que acepta secuencias de un número impar de símbolos 1.

En particular, notemos que para todo conjunto finito de enteros no negativos σ siempre existe un autómata determinístico de estados finitos que acepta una cadena de n caracteres 1 consecutivos si y sólo si $n \in \sigma$. En efecto, si llamamos m el elemento máximo de σ , podríamos considerar el autómata $M_\sigma = (Q, \{1\}, \delta, q_0, F)$ dado por: $Q = \{s_0, \dots, s_{m+1}\}$, $q_0 = s_0$, $F = \{s_i : i \in \sigma\}$, y $\delta(s_i, 1) = s_{i+1}$ para todo $i \in \llbracket 0, m \rrbracket$ y $\delta(s_{m+1}, 1) = s_{m+1}$. Si ponemos restricciones a las cadenas que el autómata recibe como entrada (por ejemplo, una cota superior a su longitud), es posible que existan otros autómatas equivalentes (en el sentido de que aceptan las mismas cadenas) y con menor cantidad de estados. Por ejemplo, si tenemos que σ es el conjunto de los números impares entre 1 y 11, y que las cadenas de entrada tienen a lo sumo 11 elementos, entonces el autómata M_i (definido en el primer ejemplo) acepta las mismas cadenas que M_σ . Este hecho nos será útil más adelante en este trabajo.

1.3. Nociones algebraicas básicas

Una *operación binaria cerrada* (o *interna*) en un conjunto S es una función $\star: S \times S \rightarrow S$. Es usual escribir $\star(s_1, s_2)$ como $s_1 \star s_2$. Tal operación es *conmutativa* si $s_1 \star s_2 = s_2 \star s_1$ para todo $s_1, s_2 \in S$, y es *asociativa* si $(s_1 \star s_2) \star s_3 = s_1 \star (s_2 \star s_3)$ para todo $s_1, s_2, s_3 \in S$. Un elemento $e \in S$ es *neutro* si $e \star s = s \star e = s$ para todo $s \in S$. A este elemento también se lo llama *identidad*. Un elemento $a \in S$ es *absorbente* si $a \star s = s \star a = a$ para todo $s \in S$. Es fácil probar que si $s \in S$ es un elemento neutro (resp. absorbente), entonces éste es único. Una operación conmutativa y asociativa $\star$ se puede extender naturalmente a cualquier subconjunto finito no vacío de S , escribiendo $\star_{x \in X} P(x)$ cuando $\{P(x) : x \in X\} \subseteq S$ y X es finito y no vacío, más aún, si la operación también tiene un elemento neutro e entonces podemos definir $\star_{x \in \emptyset} P(x) = e$.

Sean S un conjunto y $\star$ una operación binaria cerrada en S . Entonces $(S, \star)$ es un *monoide* si $\star$ es asociativa y tiene un elemento neutro. Si $\star$ es también conmutativa entonces $(S, \star)$ es un *monoide conmutativo*.

Se llama $\mathbb{F}_2$ al *cuerpo de dos elementos*, es decir, a $(\{0, 1\}, +, \cdot)$ donde $+$ y $\cdot$ son dos operaciones binarias cerradas en $\{0, 1\}$ definidas de la siguiente manera:

+	0	1
0	0	1
1	1	0

·	0	1
0	0	0
1	0	1

Una *relación binaria* en un conjunto S es un subconjunto $\mathcal{R}$ del producto cartesiano $S \times S$. Es usual escribir $(s_1, s_2) \in \mathcal{R}$ como $s_1 \mathcal{R} s_2$. Decimos que $\mathcal{R}$ es *reflexiva* si $s \mathcal{R} s$ para todo $s \in S$, *antisimétrica* si $s_1 \mathcal{R} s_2 \wedge s_2 \mathcal{R} s_1 \Rightarrow s_1 = s_2$ para todo $s_1, s_2 \in S$, y *transitiva* si $s_1 \mathcal{R} s_2 \wedge s_2 \mathcal{R} s_3 \Rightarrow s_1 \mathcal{R} s_3$ para todo $s_1, s_2, s_3 \in S$. Si $\mathcal{R}$ es reflexiva, antisimétrica y transitiva, entonces $(S, \mathcal{R})$ se llama *orden parcial* (o *conjunto parcialmente ordenado*). Si además $s_1 \mathcal{R} s_2 \vee s_2 \mathcal{R} s_1$ para todo $s_1, s_2 \in S$, entonces $(S, \mathcal{R})$ se llama *orden total* (o *conjunto totalmente ordenado*).

Sea $(S, \preceq)$ un conjunto totalmente ordenado. Un *elemento máximo* es un elemento $m \in S$ tal que $s \preceq m$ para todo $s \in S$. Nótese que no todo conjunto totalmente ordenado tiene un elemento máximo, y es fácil probar que si tiene un elemento máximo entonces éste es único. La *operación mínimo*, $\min$, es la operación binaria cerrada en S tal que $\min(s_1, s_2) = s_1$ si $s_1 \preceq s_2$ y $\min(s_1, s_2) = s_2$ si $s_2 \preceq s_1$. Es fácil ver que $\min$ es conmutativa y asociativa.

Notamos $\mathbb{R}$ al conjunto de los números reales, $\mathbb{Q}$ al conjunto de los números racionales, $\mathbb{Z}$ al conjunto de los números enteros y $\mathbb{N}$ al conjunto de los enteros no negativos. Además, $\mathbb{N}^{+\infty} = \mathbb{N} \cup \{+\infty\}$ y $\mathbb{N}^{-\infty} = \mathbb{N} \cup \{-\infty\}$ (y análogamente para $\mathbb{Z}$, $\mathbb{Q}$ y $\mathbb{R}$). Notamos $\llbracket a, b \rrbracket$, con $a, b \in \mathbb{Z}$ y $a < b$, al conjunto de todos los números enteros mayores o iguales a a y menores o iguales a b , es decir, al conjunto $\{a, a+1, \dots, b\}$. Un conjunto de números naturales es *co-finito* si su complemento (respecto al conjunto de los números naturales) es finito.

A lo largo de este trabajo, utilizaremos el conjunto $\text{BOOL} = \{\text{TRUE}, \text{FALSE}\}$ de valores booleanos y todos los operadores lógicos usuales, tales como $\neg$, $\wedge$, $\vee$ y $\Rightarrow$, así como también los cuantificadores $\forall, \exists$. Definimos $\mathbb{1}: \text{BOOL} \rightarrow \{0, 1\}$ como la función tal que $\mathbb{1}(\text{TRUE}) = 1$ y $\mathbb{1}(\text{FALSE}) = 0$.

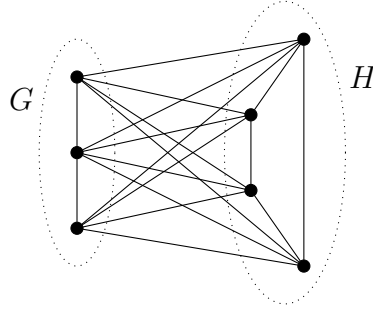
Para todo conjunto S , $\mathcal{P}(S)$ denota al conjunto de todos los subconjuntos de S . Dado un entero positivo q , una q -tupla es una tupla ordenada de q elementos y una q -partición de S es una q -tupla $(S_1, \dots, S_q)$ donde $\bigcup_{i=1}^q S_i = S$ y $S_i \cap S_j = \emptyset$ para todo $i \neq j$. Notamos $\text{part}_q(S)$ al conjunto de todas las q -particiones de S . Dado un entero no negativo k , notamos $\text{part}_q(k)$ al conjunto de todas las q -tuplas de enteros no negativos tales que la suma de todos sus elementos es igual a k . Para $X \in \text{part}_q(S)$, notamos $|X|$ a la q -tupla con $|X|_i = |X_i|$. Si $X \in \text{part}_q(S)$ y S' es otro conjunto, notamos $S' \cap X$ a la q -tupla con $(S' \cap X)_i = S' \cap X_i$. Si S y S' son conjuntos disjuntos, $X \in \text{part}_q(S)$ y $X' \in \text{part}_q(S')$ entonces notamos $X \cup X'$ a la q -tupla con $(X \cup X')_i = X_i \cup X'_i$, la cual es una partición de $S \cup S'$. Si S es un conjunto finito entonces una S -tupla es una función de dominio S , y escribimos $T = (T_x)_{x \in S}$ para indicar que T es una S -tupla para la cual cada $x \in S$ es asignado a T_x .

Sea $f: A \rightarrow B$ una función y sea $S \subseteq A$. Cuando $B \subseteq \mathbb{R}$, el *peso en S* de la función f se define como $f(S) = \sum_{s \in S} f(s)$. Notamos $f|_S$ a la función f restringida al dominio S , esto es, la función $f|_S: S \rightarrow B$ se define como $f|_S(s) = f(s)$ para todo $s \in S$. Sea A' un conjunto tal que $A \cap A' = \emptyset$, y sea $g: A' \rightarrow B$. Notamos $f \cup g$ a la función $h: A \cup A' \rightarrow B$ tal que $h(x) = f(x)$ si $x \in A$, y $h(x) = g(x)$ si $x \in A'$. Notemos que, como A y A' son conjuntos disjuntos, la función $f \cup g$ está bien definida.

1.4. Conjuntos de pesos

Sea $(\text{WEIGHTS}, \preceq)$ un conjunto totalmente ordenado con un elemento máximo (llamado **ERROR**), junto con la operación mínimo del orden $\preceq$ (llamada $\min$) y una operación binaria cerrada en **WEIGHTS** (llamada $\oplus$) que es conmutativa y asociativa, tiene un elemento neutro $I_\oplus$ y un elemento absorbente que es igual a **ERROR**, y la siguiente propiedad se satisface: $s_1 \preceq s_2 \Rightarrow s_1 \oplus s_3 \preceq s_2 \oplus s_3$ para todos $s_1, s_2, s_3 \in \text{WEIGHTS}$. En tal caso, decimos que $(\text{WEIGHTS}, \preceq, \oplus)$ es un *conjunto de pesos* (o *conjunto de costos*).

Un ejemplo de un conjunto de pesos es $(\mathbb{N}^{+\infty}, \leq, +)$. Notemos que el elemento máximo es $+\infty$ en este caso. También podríamos considerar el orden inverso de los pesos naturales: $(\mathbb{N}^{-\infty}, \geq, +)$, donde el máximo elemento ahora es $-\infty$. Otro ejemplo

Figura 1.1: Join entre dos grafos G y H .

simple digno de mencionar es $(\{0, 1\}, \leq, \text{máx})$.

1.5. Definiciones básicas de teoría de grafos

A menos que se indique lo contrario, todos los grafos considerados en este trabajo son finitos, simples (no tienen aristas dobles ni bucles) y no dirigidos. Dado un grafo G , notamos $V(G)$ al conjunto de vértices y $E(G)$ al conjunto de aristas de G . Dos vértices $u, v \in V(G)$ son *adyacentes* (o *vecinos*) si $uv \in E(G)$, y en tal caso decimos que u y v son los *extremos* de la arista uv . La *matriz de adyacencia* de G es la matriz $M \in \{0, 1\}^{|V(G)| \times |V(G)|}$, indexada por los vértices de G , con $M[u, v] = 1$ si y sólo si $uv \in E(G)$. La *vecindad abierta* de un vértice v es $N_G(v) = \{u : uv \in E(G)\}$, es decir, es el conjunto de vértices de G adyacentes a v . La *vecindad cerrada* se define como $N_G[v] = N_G(v) \cup \{v\}$. Para todo $S \subseteq V(G)$, definimos $N_G[S] = \bigcup_{v \in S} N_G[v]$. El *grado* de un vértice v es $d_G(v) = |N_G(v)|$ (también notado $\deg(v)$), es decir, es la cantidad de vecinos de v . El máximo grado entre los vértices de G se nota $\Delta(G)$.

Un grafo H es un *subgrafo* del grafo G si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$, de manera que si $uv \in E(H)$ entonces $u, v \in V(H)$. En particular, H es un *subgrafo inducido* de G si $V(H) \subseteq V(G)$ y $E(H) = \{uv : u \in V(H) \wedge v \in V(H) \wedge uv \in E(G)\}$. Si S es un subconjunto de $V(G)$, notamos $G[S]$ al subgrafo inducido de G cuyo conjunto de vértices es S .

Una *clase de grafos* es una familia de grafos que es cerrada bajo isomorfismo. Dada una clase de grafos $\mathcal{G}$ y un parámetro p , decimos que $\mathcal{G}$ tiene *el parámetro p acotado* si $\sup\{p(G) : G \in \mathcal{G}\} < \infty$.

La *unión disjunta* de dos grafos G y H es el grafo $G \cup H$ (también llamado $G \oplus H$) tal que $V(G \cup H) = V(G) \cup V(H)$ y $E(G \cup H) = E(G) \cup E(H)$. La *unión completa* (o *join*) entre dos grafos G y H es el grafo $G \nabla H$ obtenido de la unión disjunta de G y H al agregarle todas las aristas que conectan un vértice de G con un vértice de H (ver Figura 1.1). Un grafo es un *cografo* si puede construirse a partir de grafos de un sólo vértice mediante la aplicación de las operaciones join y unión disjunta.

Un *camino en un grafo* G es una sucesión $(v_1, \dots, v_k)$ de vértices distintos de G , donde $v_i v_{i+1} \in E(G)$ para todo $i \in \llbracket 1, k-1 \rrbracket$. Un grafo G es *conexo* si para todo par de vértices distintos $u, v \in V(G)$ existe un camino $(v_1, \dots, v_k)$ en G tal que $v_1 = u$ y $v_k = v$. Una *componente conexa* de un grafo G es un subgrafo conexo maximal de G . Notamos $cc(G)$ al conjunto de componentes conexas de G . Definimos $ccut(G) = \text{part}_2(cc(G))$ y llamamos *cortes conexos* a los elementos de $ccut(G)$. Para un conjunto $S \subseteq V(G)$ y cuando G queda claro por contexto, notamos $ccut(S)$ a $ccut(G[S])$.

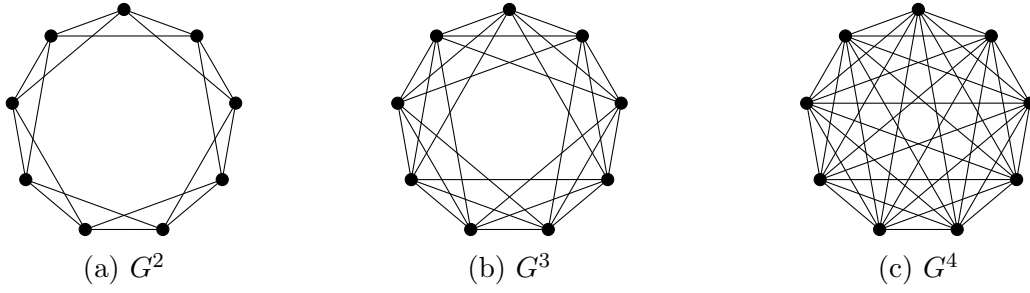


Figura 1.2: Distintas potencias de un grafo G que es un ciclo de 9 vértices.

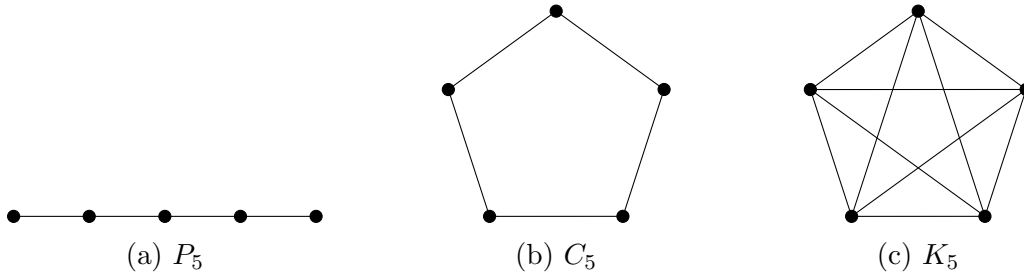


Figura 1.3: Grafos camino, ciclo y completo de 5 vértices.

Dados dos vértices u, v en un grafo G , notamos $\text{dist}_G(u, v)$ a la *distancia entre u y v* , es decir, la longitud (número de aristas) de un camino mínimo entre u y v en G . Si tal camino no existe (porque u y v están en diferentes componentes conexas de G), se considera $\text{dist}_G(u, v) = \infty$. El *diámetro* de G , $\text{diam}(G)$, es la mayor distancia entre dos vértices de G , es decir, $\text{diam}(G) = \max_{u, v \in V(G)} \text{dist}_G(u, v)$.

Sea r un entero positivo. La r -ésima *potencia* de un grafo G , notada G^r , es el grafo con $V(G^r) = V(G)$ y tal que para todo par de vértices distintos $u, v \in V(G)$, u es vecino de v en G^r si y sólo si $\text{dist}_G(u, v) \leq r$ (es decir, $E(G^r) = \{uv : \text{dist}_G(u, v) \leq r\}$). La Figura 1.2 muestra un ejemplo. Observemos que si G es conexo entonces $G^{\text{diam}(G)}$ es un grafo completo.

Notamos $N_G^r[v]$ al conjunto de vértices a distancia a lo sumo r de v en G . Una propiedad que nos será útil más adelante es que $N_G^r[v] = N_{G^r}[v]$. El grafo G^r se puede obtener en tiempo $O(|V(G)| \cdot |E(G)|)$ simplemente buscando los vértices a distancia a lo sumo r de cada $v \in V(G)$, aplicando una *búsqueda en anchura* (BFS).

Un grafo G es un *camino* si $V(G) = \{v_1, \dots, v_k\}$ y $E(G) = \bigcup_{i=1}^{k-1} \{v_i v_{i+1}\}$. Un camino de k vértices se nota P_k (ver Figura 1.3a). Un grafo G es un *ciclo* si $V(G) = \{v_1, \dots, v_k\}$, $E(G) = \left(\bigcup_{i=1}^{k-1} \{v_i v_{i+1}\}\right) \cup \{v_k v_1\}$ y $k \geq 3$. Un ciclo de k vértices usualmente se nota C_k (ver Figura 1.3b).

Un grafo G es *completo* si $N_G[v] = V(G)$ para todo $v \in V(G)$, es decir, si todos sus vértices son vecinos entre sí. Un grafo completo de n vértices usualmente se nota K_n (ver Figura 1.3c). Los grafos completos son un caso particular de cografos. Una *clique* en un grafo G es un subgrafo completo maximal de G . El *número de clique* de G , $\omega(G)$, es el mayor tamaño de una clique en G . Un conjunto I de vértices de un grafo G es *independiente* (o *estable*) si todos sus vértices son no-adyacentes entre sí. El *número de independencia* (o *de estabilidad*), $\alpha(G)$, es el mayor cardinal de un conjunto independiente en G .

Un grafo G es *bipartito* si $V(G)$ es la unión de dos conjuntos disjuntos independientes

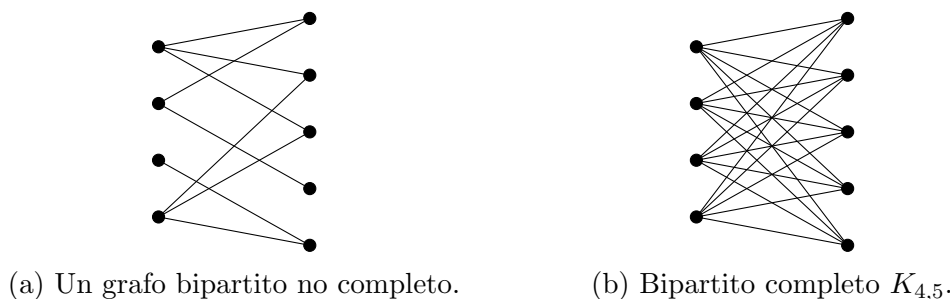


Figura 1.4: Grafos bipartitos

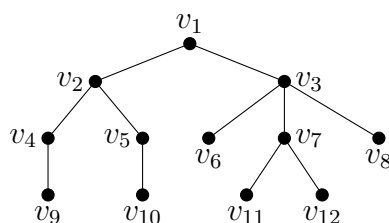


Figura 1.5: Un árbol. Si lo consideramos enraizado en el vértice v_1 entonces tenemos que, por ejemplo, v_4 es el padre de v_9 , los hijos de v_2 son v_4 y v_5 , los descendientes de v_3 son v_6 , v_7 , v_8 , v_{11} y v_{12} , y T_{v_2} es el subárbol que contiene los vértices v_2 , v_4 , v_5 , v_9 y v_{10} . Si eliminamos el vértice v_8 obtenemos un árbol binario.

V_1 y V_2 , y G es *bipartito completo* si es bipartito y todo vértice de V_1 es adyacente a todo vértice de V_2 . Notamos $K_{r,s}$ al grafo bipartito completo con $|V_1| = r$ and $|V_2| = s$. Los grafos bipartitos completos son un tipo especial de cografos. La Figura 1.4 muestra ejemplos de grafos bipartitos (uno completo y otro no completo). Si $A, B \subset V(G)$ son dos conjuntos disjuntos no vacíos de vértices, notamos $G[A, B]$ al grafo bipartito de vértices $A \cup B$ y aristas $\{ab : a \in A, b \in B \text{ y } ab \in E(G)\}$, el cual es un subgrafo de G .

Un grafo es *acíclico* si no contiene ningún ciclo de longitud al menos 3. Un grafo acíclico también se llama *bosque*. Un *árbol* es un grafo acíclico conexo. En un árbol T , es usual llamar *nodos* a los elementos de $V(T)$, y *subárbol* a un subgrafo conexo de T . Un nodo con grado a lo sumo 1 se llama *hoja*, y un nodo de grado al menos 2 se llama *nodo interno*. Un árbol está *enraizado* si un vértice fue designado como la *raíz*. Sea T un árbol enraizado en r . Para un nodo $t \in V(T) - \{r\}$, el vecino de t en el camino de t a la raíz se llama el *padre* de t . Un nodo s es un *hijo* de un nodo t si t es el padre de s . Los *descendientes* de un nodo t son los hijos de t y los descendientes de ellos (las hojas no tienen descendientes). Notamos T_t al subárbol de T inducido por el conjunto de t y sus descendientes. Un *árbol binario* es un árbol enraizado donde todo nodo interno tiene a lo sumo dos hijos. La Figura 1.5 ilustra estos conceptos.

Un grafo es una *oruga* si es un árbol tal que al quitarle sus hojas resulta en un camino. La *estrella* de n vértices, llamada S_n , es el grafo $K_{1,n-1}$ (notar que es también un árbol con $n - 1$ hojas). Ver Figura 1.6. Es fácil ver que todos los árboles son bipartitos.

Un grafo es *cordal* si todo ciclo inducido en él tiene a lo sumo 3 vértices. Un grafo G es *split* si $V(G)$ puede particionarse en un conjunto independiente y una clique (ver Figura 1.7). Directamente de su definición, los árboles y los grafos split son cordales.

Un grafo es *planar* (también llamado *plano*) si se puede representar en el plano de

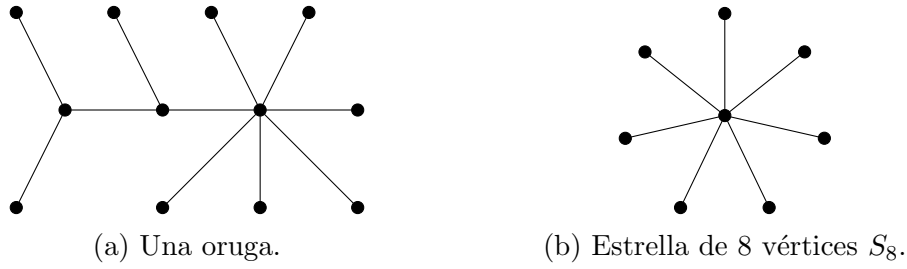
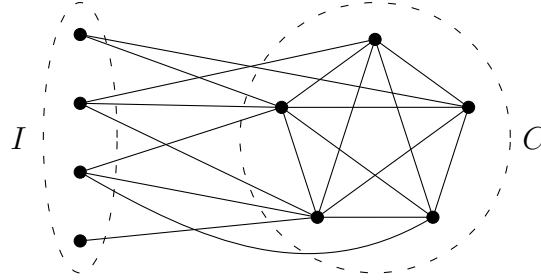


Figura 1.6: Casos particulares de árboles.

Figura 1.7: Grafo split con un conjunto independiente I y una clique C .

modo tal que sus aristas no se corten (ver Figura 1.8). La máxima cantidad de aristas que puede tener un grafo planar de n vértices es $3n + 6$ (folklore). Un grafo G es una *grilla* (también llamado *celosía*, *mall*a o *cuadrícula*) de $n \times m$ si puede ser representado en el plano como una cuadrícula de $n \times m$ puntos. Formalmente, $V(G) = \{(x, y) : x \in [1, n] \wedge y \in [1, m]\}$ y $E(G) = \{(x_1, y_1)(x_2, y_2) : |x_2 - x_1| + |y_2 - y_1| = 1\}$. Es decir, sus vértices pueden pensarse como puntos de coordenadas enteras en un rectángulo de $n \times m$ puntos, y dos vértices son vecinos si y sólo si están a distancia 1 (ver Figura 1.8b). Las grillas son planares y bipartitos de grado menor o igual a 4. Los árboles son también planares y bipartitos, pero de grado no acotado.

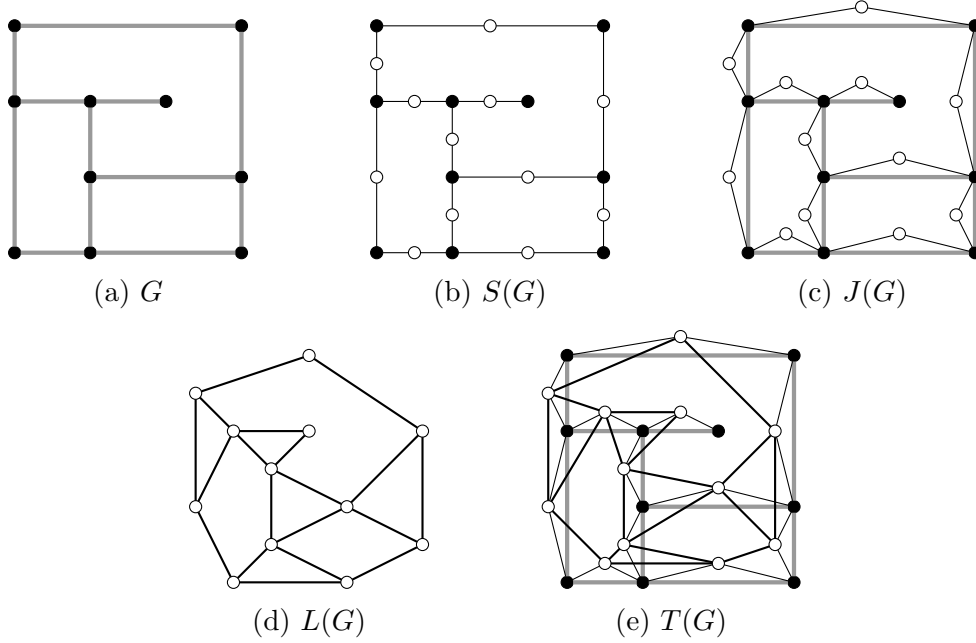
Dados conjuntos $S_1, \dots, S_n$, el *grafo de intersección* de $S_1, \dots, S_n$ es aquel grafo G con $V(G) = \{v_1, \dots, v_n\}$ tal que $v_i v_j \in E(G)$ si y sólo si $S_i \cap S_j \neq \emptyset$. Un grafo es *de intervalo* si es el grafo de intersección de intervalos en la recta real. Un grafo es *arco-circular* si es el grafo de intersección de arcos en una circunferencia. Claramente, los grafos de intervalo forman una subclase de los grafos arco-circulares. Además, los grafos de intervalo son cordales [132].

Dados un grafo G y una arista uv de G , el grafo obtenido al *subdividir* uv en G surge de G al agregar un nuevo vértice w , hacer que w sea adyacente a u y v , y eliminar la arista uv . El *grafo de subdivisión* de G , $S(G)$, es aquel que se obtiene al subdividir cada arista de G (ver Figura 1.9b). El *grafo dentado* de G , $J(G)$, es aquel que se obtiene de $S(G)$ al agregar nuevamente las aristas $E(G)$ (ver Figura 1.9c). El *grafo de línea* de G , $L(G)$, tiene como conjunto de vértices a $E(G)$ y dos vértices son adyacentes en $L(G)$ si y sólo si sus correspondientes aristas en G tienen un extremo en común (ver Figura 1.9d). El *grafo total* de G , $T(G)$, tiene como conjunto de vértices a $V(G) \cup E(G)$, y es tal que $V(G)$ induce G , $E(G)$ induce $L(G)$, y dos vértices $v \in V(G)$ y $e \in E(G)$ son adyacentes en $T(G)$ si y sólo si v es un extremo de e en G (ver Figura 1.9e).

Un *emparejamiento* (llamado *matching* en inglés) en un grafo es un conjunto de aristas que no comparten ningún extremo (ver Figura 1.10a). Un emparejamiento en



Figura 1.8: Ejemplos de grafos planares.

Figura 1.9: Un grafo G , su grafo de subdivisión $S(G)$, su grafo dentado $J(G)$, su grafo de línea $L(G)$ y su grafo total $T(G)$.

un grafo G determina un conjunto independiente en el grafo de línea de G , $L(G)$. Mediante el Algoritmo de Edmonds [74], podemos encontrar un emparejamiento máximo (es decir, con la máxima cantidad posible de aristas) en tiempo polinomial. Si un emparejamiento M en un grafo G es tal que el subgrafo de G inducido por los extremos de las aristas en M es una unión disjunta de varios grafos K_2 , entonces decimos que M es un *emparejamiento inducido* (ver Figura 1.10b). Equivalentemente, un emparejamiento inducido en G determina un conjunto independiente en el cuadrado del grafo de línea, $L(G)^2$. Determinar si existe un emparejamiento inducido de tamaño mayor o igual a k , para algún entero k dado como entrada, es un problema NP-completo, incluso en grafos bipartitos de grado cuatro [160].

Un *k -coloreo propio* de un grafo es una asignación de colores a vértices tal que todo vértice recibe un color diferente al de sus vecinos. Equivalentemente, es una partición de sus vértices en a lo sumo k conjuntos independientes. El *número cromático* de un grafo G , notado $\chi(G)$, es el menor valor de k tal que G admite un k -coloreo propio. Los grafos bipartitos son exactamente aquellos que tienen número cromático menor o igual a 2. Para todo $k \geq 3$, determinar si un grafo es k -coloreable (es decir, si admite un k -coloreo propio) es un problema NP-completo [92], mientras que para $k = 2$ se puede resolver fácilmente en tiempo lineal. Un grafo G es *perfecto* si $\chi(H) = \omega(H)$ para

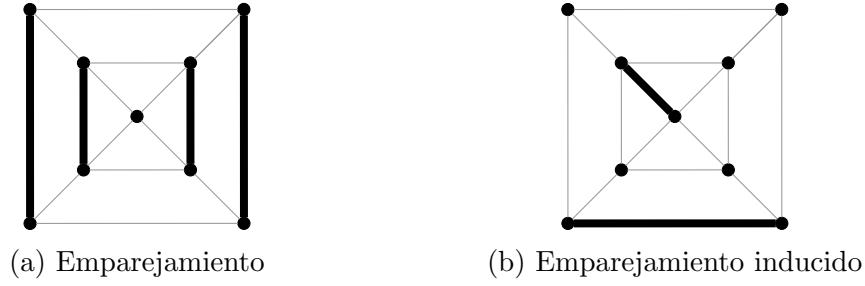


Figura 1.10: Un mismo grafo y distintos emparejamientos (destacados en línea gruesa).

todo subgrafo inducido H de G . Ejemplos de grafos perfectos incluyen los bipartitos, cografos y cordales [116].

Decimos que un par de vértices o un par de aristas se *dominan* entre sí cuando son iguales o bien adyacentes, y un vértice v y una arista e se dominan cuando v es un extremo de e . Notamos $\gamma_{U,W}(G)$, siendo U y W conjuntos de elementos del grafo G , al mínimo cardinal de un subconjunto S de U que domina a W . El parámetro $\gamma_{V(G),V(G)}(G)$ también se nota simplemente $\gamma(G)$, y se llama el *número de dominación* de G . El problema de determinar $\gamma(G)$ se conoce como el problema del MÍNIMO CONJUNTO DOMINANTE y su problema de decisión asociado es NP-completo [92]. Los parámetros $\gamma_{V,E}$ y $\gamma_{E,V}$ se asocian a los problemas del MÍNIMO CUBRIMIENTO POR VÉRTICES (en inglés, *vertex cover*) y MÍNIMO CUBRIMIENTO POR ARISTAS (en inglés, *edge cover*), respectivamente. En el problema MÍNIMA $\{k\}$ -DOMINACIÓN, dado un grafo G queremos encontrar el mínimo costo de una función $f: V(G) \rightarrow \llbracket 0, k \rrbracket$ tal que $\sum_{u \in N_G[v]} f(u) \geq k$ para todo $v \in V(G)$.

1.6. Parámetros de anchura en grafos

1.6.1. Treewidth

Una *descomposición en árbol* (*tree decomposition* en inglés) de un grafo G es un árbol T junto con una familia $\{X_t : t \in V(T)\}$ de subconjuntos de $V(G)$ (donde cada X_t es llamado *bolsa*), que satisfacen las siguientes propiedades:

- (W1) $\bigcup_{t \in V(T)} X_t = V(G)$;
- (W2) para toda arista uv en G , existe al menos un nodo $t \in V(T)$ tal que la bolsa X_t contiene tanto a u como a v ;
- (W3) para todo vértice v en G , el conjunto de nodos $\{t \in V(T) : v \in X_t\}$ induce un subárbol de T .

La *anchura* de una descomposición en árbol $(T, \{X_t\}_{t \in V(T)})$ es igual al tamaño máximo de una bolsa menos uno, es decir, $\max\{|X_t| - 1 : t \in V(T)\}$. La *anchura de árbol* (o *treewidth*) de un grafo G , notado $tw(G)$, es el mínimo w tal que G admite una descomposición en árbol de anchura igual a w . Dada una clase de grafos $\mathcal{G}$, definimos $tw(\mathcal{G}) = \sup\{tw(G) \mid G \in \mathcal{G}\}$. Decimos que $\mathcal{G}$ tiene *treewidth acotada* si $tw(\mathcal{G}) < \infty$. La *anchura de camino* (*pathwidth*, pw) se define análogamente pero exigiendo que T sea un camino.

El parámetro treewidth fue definido formalmente en 1983 por Robertson y Seymour [152], pero también fue previamente considerado bajo diferentes nombres por Bertelè y Brioschi [29] y Halin [106]. Entre las clases de grafos de treewidth acotada podemos mencionar: bosques (treewidth 1), pseudo-bosques, cactus, grafos outerplanar y grafos serie-paralelo (treewidth a lo sumo 2), grafos Halin y redes Apollonian (treewidth a lo sumo 3) [34, 45, 126, 151].

Sea k un entero positivo. Los k -árboles se definen recursivamente de la siguiente manera: el grafo completo de k vértices (K_k) es un k -árbol, y si G es un k -árbol y Q es una clique de G de k vértices, entonces también es un k -árbol el grafo G' obtenido de G al agregar un vértice $v \notin V(G)$ tal que sus vecinos son exactamente los vértices de Q ($N_{G'}(v) = Q$). Un k -árbol parcial es un subgrafo de un k -árbol. Un grafo tiene treewidth a lo sumo k si y sólo si es un k -árbol parcial [34, Teorema 35]. Luego, treewidth se puede definir alternativamente mediante k -árboles parciales: la treewidth de G es el menor entero k tal que G es un k -árbol parcial.

Proposición 1.6.1 (Folklore). *Sea k un entero positivo y sea G un grafo de treewidth k . Entonces $|E(G)| \leq k|V(G)| - \frac{k(k+1)}{2}$.*

Demostración. Como $tw(G) = k$ entonces G es un k -árbol parcial y por lo tanto es un subgrafo de un k -árbol de $|V(G)|$ vértices. Directamente de la definición de k -árbol obtenemos que la cantidad de aristas de un k -árbol de n vértices es $\frac{k(k-1)}{2} + k(n-k)$. Luego, la cantidad de aristas de G es a lo sumo $\frac{k(k-1)}{2} + k(|V(G)| - k) = k|V(G)| - \frac{k(k+1)}{2}$. $\square$

Proposición 1.6.2 (Folklore). *Si H es un subgrafo de un grafo G entonces $tw(H) \leq tw(G)$.*

Teorema 1.6.3 ([152, páginas 1 y 2]). *Para todo grafo G , $tw(G) \geq \omega(G) - 1$. Más aún, $tw(G) \geq \chi(G) - 1$.*

Teorema 1.6.4 ([126, página 76]). $tw(S(G)) = tw(G)$.

Teorema 1.6.5 ([36, Teorema 1]). *Existe un algoritmo tal que, dados un grafo G de n vértices y un entero k , en tiempo $O(c^k n)$ para alguna constante $c \in \mathbb{N}$, o bien responde que la treewidth de G es mayor a k , o bien construye una descomposición en árbol de G de anchura a lo sumo $5k + 4$.*

Para un grafo dado pueden existir múltiples descomposiciones en árbol, incluso si pedimos que tengan anchura mínima (ver Figura 1.11). En los algoritmos para resolver problemas en clases de treewidth acotada usualmente se utiliza un tipo especial de descomposiciones en árbol que provee aún más estructura. Una descomposición en árbol $(T, \{X_t\}_{t \in V(T)})$ es *agradable* [126, Definición 13.1.4] si:

- T es un árbol binario enraizado;
- si un nodo i tiene dos hijos j y k entonces $X_i = X_j = X_k$;
- si un nodo i tiene sólo un hijo j , entonces
 - $|X_i| = |X_j| - 1$ y $X_i \subset X_j$, o bien
 - $|X_i| = |X_j| + 1$ y $X_i \supset X_j$.

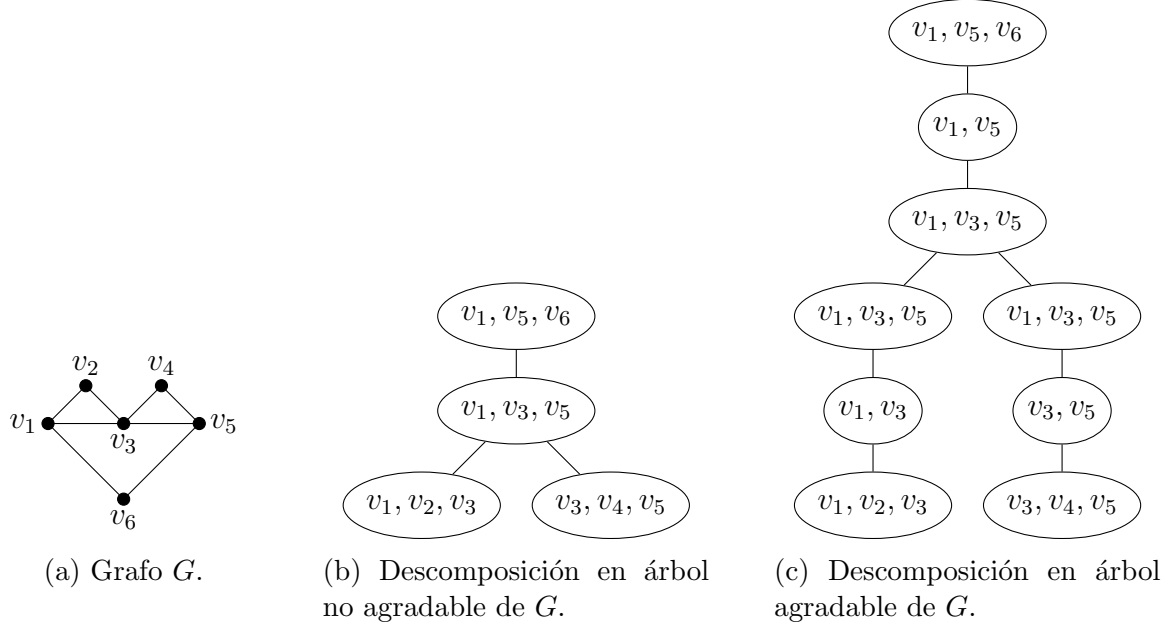


Figura 1.11: Un grafo y algunas de sus posibles descomposiciones en árbol.

Para todo nodo $i \in V(T)$, denotaremos G_i al subgrafo inducido $G \left[\bigcup_{j \in V(T_i)} X_j \right]$.

Teorema 1.6.6 ([126, Lema 13.1.3]). *Para una constante k , dada una descomposición en árbol de un grafo G , de anchura k y con $O(|V(G)|)$ nodos, se puede encontrar una descomposición en árbol agradable de G , de anchura k y a lo sumo $4|V(G)|$ nodos, en tiempo $O(|V(G)|)$.*

En esta tesis vamos a trabajar con una ligera modificación de las descomposiciones en árbol agradables, donde las bolsas de la raíz y de las hojas tienen un único vértice cada una.

Definición 1.6.7. Una descomposición en árbol $(T, \{X_t\}_{t \in V(T)})$ se llama *sencilla* si

- T es un árbol binario enraizado en r , tal que $|X_r| = 1$;
- si un nodo i tiene dos hijos j y k entonces $X_i = X_j = X_k$; (nodo *unión*)
- si un nodo i tiene sólo un hijo j , entonces
 - $|X_i| = |X_j| - 1$ y $X_i \subset X_j$, o bien (nodo *eliminación*)
 - $|X_i| = |X_j| + 1$ y $X_i \supset X_j$; (nodo *inserción*)
- si un nodo i no tiene hijos, entonces $|X_i| = 1$. (nodo *hoja*)

Es fácil ver que, dada una descomposición en árbol agradable de anchura k y $O(|V(G)|)$ nodos, se puede construir en tiempo $O(k|V(G)|)$ una descomposición en árbol sencilla de anchura k y $O(k|V(G)|)$ nodos.

Corolario 1.6.8. *Existe un algoritmo tal que, dados un grafo G de n vértices y un entero k , en tiempo $O(c^k n)$ para alguna constante $c \in \mathbb{N}$ (es decir, FPT lineal parametrizado por k), o bien responde que la treewidth de G es mayor a k , o bien construye una descomposición en árbol sencilla de G de anchura a lo sumo $5k + 4$ y a lo sumo $4kn$ nodos.*

El siguiente resultado, el cual probamos junto a Flavia Bonomo en [41], relaciona los valores de treewidth y grado de un grafo G con los de su r -ésima potencia G^r . En particular, implica que si tanto la treewidth como el grado de G están acotados por una constante entonces, para una potencia fija r , la treewidth y el grado de G^r también están acotados por una constante. Notemos que es necesario pedir que el grado esté acotado, ya que, por ejemplo, la segunda potencia de una estrella S_n (de treewidth 1) es un grafo completo (de treewidth $n - 1$).

Lema 1.6.9 ([41, Lema 6.0.2]). *Sean G un grafo y r un entero mayor o igual a 2. Luego,*

$$\Delta(G) \leq \Delta(G^r) \leq \Delta(G)^r, \text{ y}$$

$$\max(\text{tw}(G), \Delta(G)) \leq \text{tw}(G^r) \leq (\text{tw}(G) + 1)(\Delta(G) + 1)^{\lceil \frac{r}{2} \rceil} - 1.$$

Demostración. En primer lugar, la desigualdad $\Delta(G) \leq \Delta(G^r) \leq \Delta(G)^r$ se deduce directamente de la definición de potencia de un grafo. A continuación estudiaremos la segunda desigualdad.

Veamos que se cumple la cota inferior. Sea v un vértice de G de grado máximo. En G^r , el grafo inducido por $N_G[v]$ es una clique de tamaño $\Delta(G) + 1$ y, por el Teorema 1.6.3, tenemos que $\text{tw}(G^r) \geq \Delta(G)$. Como G es un subgrafo de G^r y, por la Proposición 1.6.2, tenemos $\text{tw}(G^r) \geq \text{tw}(G)$.

Demostraremos ahora que vale la cota superior. Sea $(T, \{X_t\}_{t \in V(T)})$ una descomposición en árbol de G . Para todo $t \in V(T)$, sea Y_t el conjunto de vértices que están a distancia menor o igual a $\lceil \frac{r}{2} \rceil$ de un vértice de X_t . Notemos que $X_t \subseteq Y_t$ para todo $t \in V(T)$. Probaremos que $(T, \{Y_t\}_{t \in V(T)})$ es una descomposición en árbol de G^r , y para ello probaremos que se satisfacen las tres propiedades (W1), (W2) y (W3) de la definición de treewidth.

- (W1) Como $V(G) = \bigcup_{t \in V(T)} X_t \subseteq \bigcup_{t \in V(T)} Y_t \subseteq V(G)$ y $V(G) = V(G^r)$, sigue que $\bigcup_{t \in V(T)} Y_t = V(G^r)$.
- (W2) Sean u, v dos vértices que son vecinos en G^r . Si también son vecinos en G entonces existe una bolsa X_t que los contiene a ambos y entonces, como $X_t \subseteq Y_t$, la propiedad (W2) se cumple en este caso. Caso contrario, existe un vértice w que está a distancia a lo sumo $\lceil \frac{r}{2} \rceil$ tanto de u como de v en G . Sea $t \in V(T)$ tal que la bolsa X_t contiene a w . Luego, $w \in Y_t$ (porque $X_t \subseteq Y_t$) y además $u, v \in Y_t$ (porque tanto u como v están a distancia a lo sumo $\lceil \frac{r}{2} \rceil$ de $w \in X_t$). En consecuencia, la propiedad (W2) también se cumple en este caso.
- (W3) Para todo $u \in V(G)$, sean $T_u^X = T[\{t \in V(T) : u \in X_t\}]$ y $T_u^Y = T[\{t \in V(T) : u \in Y_t\}]$. Aplicando la propiedad (W3) a $(T, \{X_t\}_{t \in V(T)})$, obtenemos que T_u^X es un subárbol de T para todo $u \in V(G)$. Sea $v \in V(G)$. Vamos a probar que T_v^Y es conexo. Sea $t \in V(T_v^Y)$. Notemos que para probar que T_v^Y es conexo es suficiente probar que existe un camino en T_v^Y entre t y todo nodo $s \in V(T_v^Y) - V(T_v^X)$. Sea $s \in V(T_v^Y) - V(T_v^X)$. Por definición de las bolsas Y , sabemos que $v \in Y_{t'}$ si y sólo si $t' \in V(T_v^X)$ o $t' \in V(T_u^X)$ para algún otro vértice u tal que $\text{dist}_G(v, u) \leq \lceil \frac{r}{2} \rceil$. Por lo tanto, existe un vértice $v_s \in Y_s$ distinto de v tal que $\text{dist}_G(v, v_s) \leq \lceil \frac{r}{2} \rceil$. Además, todo vértice u en el camino mínimo entre v y v_s cumple que $V(T_u^X) \subseteq V(T_v^Y)$. Como T_u^X es conexo para todo $u \in V(G)$ y $V(T_u^X) \cap V(T_w^X) \neq \emptyset$ para toda arista

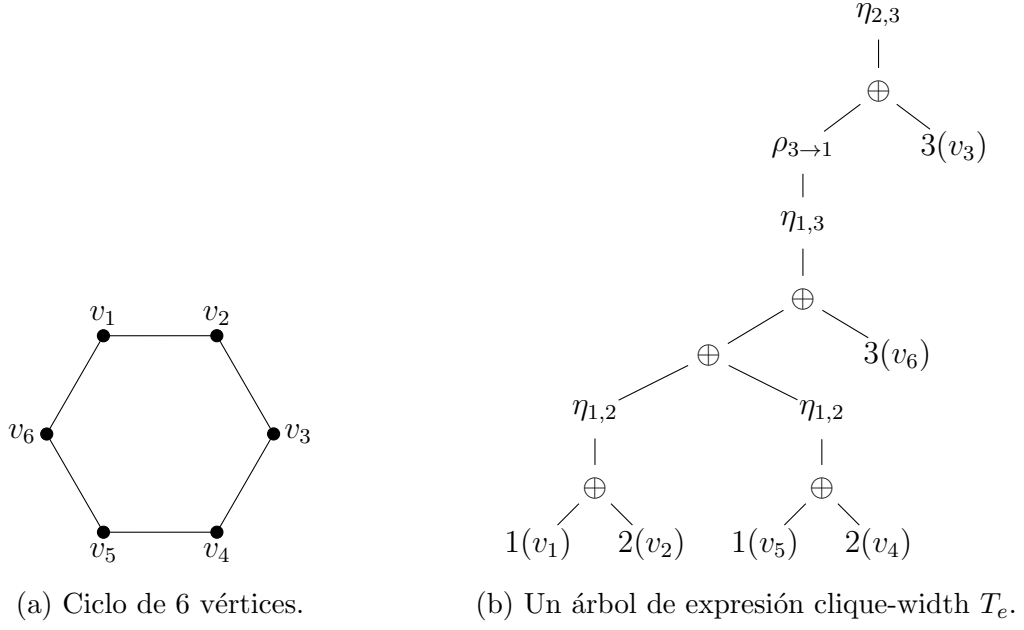


Figura 1.12: Un grafo y un árbol de expresión clique-width asociado.

$uw \in E(G)$ (por la propiedad (W2) aplicada a $(T, \{X_t\}_{t \in V(T)})$ y la arista uw), entonces existe un camino en T_v^Y entre t y s . Por lo tanto la propiedad (W3) vale para $(T, \{Y_t\}_{t \in V(T)})$.

Como cada bolsa Y_t tiene a lo sumo $(tw(G) + 1)(\Delta(G) + 1)^{\lceil \frac{r}{2} \rceil}$ vértices, obtenemos $tw(G^r) \leq (tw(G) + 1)(\Delta(G) + 1)^{\lceil \frac{r}{2} \rceil} - 1$. $\square$

1.6.2. Clique-width

La *anchura de clique* (o *clique-width*) de un grafo G , notada $cw(G)$, fue presentada por Courcelle, Engelfriet y Rozenberg en [66]. Se define como el mínimo número de etiquetas necesarias para construir G usando las siguientes cuatro operaciones:

- creación de un nuevo vértice v con etiqueta i (notado $i(v)$);
- unión disjunta de dos grafos etiquetados G_1 y G_2 (notado $G_1 \oplus G_2$);
- unión completa entre dos etiquetas i y j , $i \neq j$, es decir, adición de una arista entre cada vértice con etiqueta i y cada vértice con etiqueta j (notado $\eta_{i,j}$);
- renombre de la etiqueta i a la etiqueta j , es decir, asignación de la etiqueta j a cada vértice con etiqueta i (notado $\rho_{i \rightarrow j}$).

Una *expresión clique-width* no es más que una expresión bien formada usando las cuatro operaciones mencionadas arriba. Para una expresión e , notamos G_e al grafo determinado por e . Si el número de etiquetas distintas usadas en una expresión e es a lo sumo k , decimos que es una k -*expresión clique-width* (o simplemente k -*expresión*).

Ejemplo 1.6.10. El grafo G ilustrado en la Figura 1.12a puede ser construido usando la siguiente 3-expresión, y por lo tanto el grafo G tiene clique-width a lo sumo 3.

$$\eta_{2,3}(\rho_{3 \rightarrow 1}(\eta_{1,3}((\eta_{1,2}(1(v_1) \oplus 2(v_2)) \oplus \eta_{1,2}(1(v_5) \oplus 2(v_4))) \oplus 3(v_6)) \oplus 3(v_3))).$$

El siguiente resultado de Fomin y Korhonen nos dice que si sabemos que una clase de grafos tiene clique-width a lo sumo k entonces en tiempo cuadrático podemos encontrar expresiones “lo suficientemente buenas” para dichos grafos, y por ello nos referimos a expresiones donde la cantidad de etiquetas sólo dependa de k y no de la cantidad de vértices. En ese sentido, si k es una constante, entonces obtenemos expresiones clique-width con cantidad de etiquetas acotada por una constante.

Teorema 1.6.11 ([88, Corolario 1.1]). *Para un entero k y un grafo G de n vértices, existe un algoritmo que, en tiempo $2^{2^{O(k)}} n^2$, o bien computa una $(2^{2k+1} - 1)$ -expresión clique-width de G , o bien concluye que la clique-width de G es mayor a k .*

Una k -expresión clique-width es *irredundante* si cada vez que se ejecuta una operación de unión completa $\eta_{i,j}$ no hay ninguna arista previamente existente entre vértices con etiqueta i y vértices con etiqueta j . Nótese que la expresión mencionada en el Ejemplo 1.6.10 es irredundante. Courcelle y Olariu [69] probaron que cualquier grafo G que admite una k -expresión también admite una k -expresión irredundante.

A toda expresión clique-width e le podemos asociar naturalmente un árbol binario enraizado T_e , donde cada nodo indica una de las posibles operaciones (ver Figura 1.12b como ejemplo). Las hojas siempre indican una operación de creación de un nuevo vértice v con etiqueta i , $i(v)$, y los únicos nodos con dos hijos son aquellos que indican una unión disjunta, $\oplus$. Notemos que, para todo nodo $t \in V(T_e)$, el subárbol de T_e con raíz en t define una expresión e_t que se corresponde con el grafo G_{e_t} que es a su vez un subgrafo de G_e . Decimos que e' es una *subexpresión* de e si e' es una expresión determinada por el subárbol de T_e con raíz en algún nodo $t \in V(T_e)$.

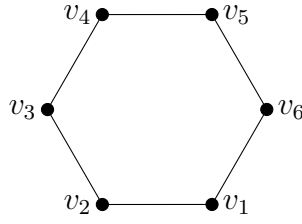
Consideremos cualquier vértice v en G_{e_t} para algún $t \in V(T_e)$. Luego, todos los vecinos de v en G_e que aún no son vecinos de v en G_{e_t} (es decir, aquellos vértices cuyas aristas con v se definen en operaciones asociadas a nodos que son ancestros de t en T_e) serán llamados *próximos vecinos de v respecto a e_t* . Notemos que para cualquier par de vértices en G_{e_t} que tengan la misma etiqueta, sus conjuntos de próximos vecinos respecto a e_t son idénticos.

Dada una k -expresión e , definimos la función $\ell_e : V(G_e) \rightarrow \llbracket 1, k \rrbracket$ tal que $\ell_e(v)$ es la etiqueta final de v , es decir, la etiqueta que tiene v después de la operación correspondiente a la raíz de T_e . También definimos $\ell(e) = \{\ell_e(v) : v \in V(G_e)\}$ y $\overline{\ell(e)} = \llbracket 1, k \rrbracket - \ell(e)$, es decir, es el conjunto de etiquetas i para las cuales no existe ningún $v \in V(G_e)$ tal que $\ell_e(v) = i$.

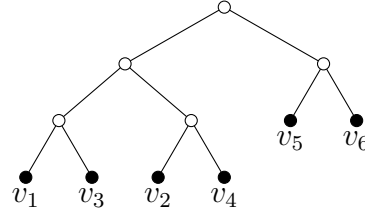
En lo que resta de este trabajo, consideraremos únicamente expresiones irredundantes donde en cada operación de renombre $\rho_{i \rightarrow j}(e)$ tenemos $j \in \ell(e)$. Nótese que, bajo estas suposiciones, el número total de operaciones en una expresión clique-width e de un grafo G pertenece a $O(|V(G)| + |E(G)|)$, ya que se realizan $|V(G)|$ operaciones de creación, $|V(G)| - 1$ uniones disjuntas, a lo sumo $|E(G)|$ uniones completas y a lo sumo $|V(G)|$ renombres de etiqueta.

1.6.3. Árboles binarios de descomposición: carvingwidth, mim-width y otros parámetros

Dado un grafo G , un árbol binario T donde cada nodo interno tiene exactamente 2 hijos junto con una biyección δ entre los vértices de G y las hojas de T , se conoce como un *árbol binario de descomposición* de G . Sea (T, δ) un árbol binario de descomposición



(a) El ciclo de 6 vértices.



(b) Un árbol binario de descomposición.

Figura 1.13: Un grafo G y un árbol binario de descomposición (T, δ) asociado a G .

de un grafo G . Para todo $t \in V(T)$, notamos T_t al conjunto de vértices de G que se corresponden (mediante δ) con las hojas que son t o descendientes de t en T . También notamos $T_{\bar{t}} = V(G) - T_t$. Ver Figura 1.13.

Para una función $f : \mathcal{P}(V(G)) \rightarrow \mathbb{R}$ tal que $f(S) = f(V(G) - S)$ para todo subconjunto $S \subseteq V(G)$, la *anchura f* (o *f -width*) de un árbol binario de descomposición (T, δ) se define como el máximo valor de $f(T_t)$ con $t \in V(T)$. La *f -width de un grafo G* es el menor valor de la f -width de (T, δ) entre todos los árboles binarios de descomposición (T, δ) de G [170, Definición 3.1.3].

De esta definición general se desprenden, entre otras, las siguientes definiciones concretas de parámetros de anchura en grafos:

- **carvingwidth (carw)** [155, 156]: $f(S)$ es la cantidad de aristas del grafo bipartito $G[S, V(G) - S]$.
- **mm-width (mmw)** [170]: $f(S)$ es la cantidad de aristas de un matching máximo (no necesariamente inducido) en el grafo bipartito $G[S, V(G) - S]$.
- **mim-width (mimw)** [170]: $f(S)$ es la cantidad de aristas de un matching inducido máximo en el grafo bipartito $G[S, V(G) - S]$.
- **sim-width (simw)** [124]: $f(S)$ es la cantidad máxima de aristas de un matching inducido en el grafo G que tienen un extremo en S y el otro en $V(G) - S$.
- **rankwidth (rw)** [117]: si llamamos M a la matriz de adyacencia en $\mathbb{F}_2$ de G , entonces $f(S)$ es el rango de la submatriz de M inducida por las filas correspondientes a S y las columnas correspondientes a $V(G) - S$.

Ejemplo 1.6.12. A modo de ejemplo, veamos la Figura 1.13. Si llamamos t al nodo padre de la hoja asociada a v_4 , entonces $T_t = \{v_2, v_4\}$ y $T_{\bar{t}} = \{v_1, v_3, v_5, v_6\}$. Para este t tenemos $\text{mim}(T_t) = 2$ (el matching inducido dado por v_1v_2 y v_4v_5). Como para los demás nodos no obtenemos un valor mayor, tenemos que $\text{mimw}(T, \delta) = 2$. Por otro lado, se puede ver muy fácilmente que no existe ninguna descomposición de anchura 1, y por lo tanto $\text{mimw}(G) = 2$.

Calcular la carvingwidth de un grafo es un problema NP-difícil [156] pero determinar si un grafo tiene carvingwidth a lo sumo k es FPT parametrizado por k [166]. Se sabe que calcular la mim-width de un grafo es un problema NP-difícil y además W[1]-difícil parametrizado por mim-width [163] pero para algunas clases de grafos (por

ejemplo, grafos arco-circulares) podemos calcular en tiempo polinomial un árbol binario de descomposición de *mim-width* acotada por una constante [23, 49]. Sin embargo, no se sabe si es posible obtener tal descomposición en tiempo polinomial en cualquier clase de grafos de *mim-width* acotada por una constante. Es más, tampoco se sabe aún si, para una constante w , se puede obtener en tiempo polinomial un árbol de descomposición de *mim-width* $f(w)$, cualquiera sea la función f , de un grafo de *mim-width* a lo sumo w [144].

Si en lugar de un árbol binario de descomposición consideramos un orden de los vértices, obtenemos la “versión lineal” de estos parámetros. Formalmente, dado un grafo G y una función $f : \mathcal{P}(V(G)) \rightarrow \mathbb{R}$, la *anchura f lineal* (o *f -width lineal*) de un ordenamiento $v_1, \dots, v_n$ de los vértices de G , se define como el máximo valor de $f(V_i)$ con $i \in [1, n]$, siendo $V_i = \{v_1, \dots, v_i\}$. La *anchura f lineal* de G es el menor valor de la anchura f lineal entre todos los ordenamientos de los vértices de G . Es equivalente a considerar árboles binarios de descomposición que sean orugas [170, Definición 3.1.5]. De esta manera se pueden definir los siguientes parámetros de anchura en grafos:

- **cutwidth (cutw) [59]:** $f(S)$ es la cantidad de aristas de $G[S, V(G) - S]$. Podríamos decir que este parámetro es la “carvingwidth lineal”, aunque nunca se lo menciona con ese nombre.
- **mim-width lineal (lmimw) [170]:** $f(S)$ es la cantidad de aristas de un matching inducido máximo en $G[S, V(G) - S]$.
- **sim-width lineal (lsimw) [124]:** $f(S)$ es la cantidad máxima de aristas de un matching inducido en G que tienen un extremo en S y el otro en $V(G) - S$.

1.6.4. Relaciones entre los parámetros

Para dos parámetros p y q , notamos $p \preceq q$ si existe una función $f : \mathbb{N} \rightarrow \mathbb{N}$ tal que $p(G) \leq f(q(G))$ para todo grafo G . En particular, para las clases de grafos donde q está acotado por alguna constante tenemos que p también está acotado por alguna constante. Si $p \preceq q$ y $q \preceq p$, lo notamos $p \approx q$. El siguiente resultado muestra la importancia de conocer este tipo de relaciones entre parámetros desde el punto de vista de la complejidad parametrizada. Fue presentado por Sasák en [155] para problemas FPT, pero su demostración puede adaptarse fácilmente a XP y para-NP-h.

Proposición 1.6.13 (Adaptación de [155, Teorema 1.1]). *Sean p y q dos parámetros tales que $p \preceq q$. Si un problema Π es FPT (resp. XP) parametrizado por p entonces Π es FPT (resp. XP) parametrizado por q . Por otro lado, si Π es para-NP-difícil parametrizado por q entonces Π es para-NP-difícil parametrizado por p .*

A continuación resumiremos las relaciones que se conocen entre los parámetros de anchura definidos en la sección anterior y probaremos una nueva desigualdad que involucra *sim-width*, grado máximo y *carvingwidth*. Lo primero que podemos observar, directamente a partir de la definición, es que $carw \preceq cutw$.

Lema 1.6.14 ([22, Observación 1]). $\Delta \preceq carw$. Más aún, $\Delta(G) \leq carw(G)$ para todo grafo G .

Lema 1.6.15 ([34, Teorema 49]). *Las clases de grafos de pathwidth y grado acotados son de cutwidth acotada, y viceversa.*

Corolario 1.6.16. *Las clases de grafos de pathwidth y carvingwidth acotadas son de cutwidth acotada, y viceversa.*

Lema 1.6.17 ([170, Teorema 4.2.5]). $tw \approx mmw$. Más aún, para todo grafo G , $\frac{1}{3}(tw(G) + 1) \leq mmw(G) \leq tw(G) + 1$.

Lema 1.6.18 ([155, Teorema 5.5]). $tw \preceq carw$.

Lema 1.6.19 ([65, Teorema 1.2]). $cw \preceq tw$. Más aún, $cw(G) \leq 3 \cdot 2^{tw(G)-1}$ para todo grafo G .

Lema 1.6.20 ([117, Proposición 6.3]). $cw \approx rw$. Más aún, $rw(G) \leq cw(G) < 2^{rw(G)+1}$ para todo grafo G .

Lema 1.6.21 ([170, Teoremas 4.2.9 y 4.2.10]). $mimw \preceq cw$. Más aún, para todo grafo G , $mimw(G) \leq cw(G)$.

Por definición, $tw(G) \leq pw(G)$ para todo grafo G y por lo tanto $tw \preceq pw$, pero existen clases de grafos de treewidth acotada y pathwidth no acotada (por ejemplo, los árboles binarios tienen treewidth 1 pero pathwidth no acotada [155]). Como indica el Lema 1.6.19, toda clase de grafos de treewidth acotada es también de clique-width acotada, pero esto no es cierto a la inversa: por ejemplo, los grafos completos tienen clique-width 2 y treewidth no acotada ($tw(K_n) = n - 1$ por el Teorema 1.6.3). Todas las clases de clique-width acotada tienen mim-width acotada (Lema 1.6.21), pero existen clases (por ejemplo, los grafos de intervalo) de mim-width 1 y clique-width no acotada [99, 170]. Directamente de la definición, sabemos que la mim-width es mayor o igual a la sim-width, pero existen clases (por ejemplo, los grafos cordales) de sim-width 1 y mim-width no acotada [124].

A continuación presentaremos un nuevo resultado relacionando sim-width, grado máximo y carvingwidth. Este resultado fue obtenido en colaboración con Felix Mann y no ha sido publicado aún. Antes de enunciar el teorema, presentaremos un concepto que utilizaremos en la demostración. Un *coloreo fuerte de aristas* (en inglés, *strong edge-coloring*) de un grafo es un coloreo de las aristas tal que cada clase de color conforma un matching inducido del grafo. El *índice cromático fuerte* de un grafo G , $\chi^s(G)$, es el menor número de colores que se necesitan para obtener un tal coloreo de G . Se sabe que existe una constante no negativa c tal que $\chi^s(G) \leq (2 - c)\Delta(G)^2$ para todo grafo G [143].

Teorema 1.6.22. *Sea G un grafo de sim-width k y grado máximo Δ . Entonces $carw(G) \leq 2k\Delta^2$.*

Demostración. Consideremos un árbol binario de descomposición (T, δ) de sim-width k . Por definición de sim-width tenemos que, para todo $t \in V(T)$, $sim(T_t) \leq k$. Es decir, k es una cota superior a la cantidad de aristas de un matching inducido en G que tienen un extremo en T_t y el otro en $T_{\bar{t}}$. Luego, en cualquier coloreo fuerte de G no podemos tener más de k aristas de un mismo color entre vértices en T_t y $T_{\bar{t}}$ (de lo contrario tendríamos un matching de más de k aristas y por lo tanto un valor mayor de sim-width). Por lo tanto la cantidad de aristas que tienen un extremo en T_t y el otro en $T_{\bar{t}}$ no puede ser mayor a $k\chi^s(G)$. Así, como $\chi^s(G) \leq 2\Delta^2$ [143], obtenemos $carw(G) \leq 2k\Delta^2$. $\square$

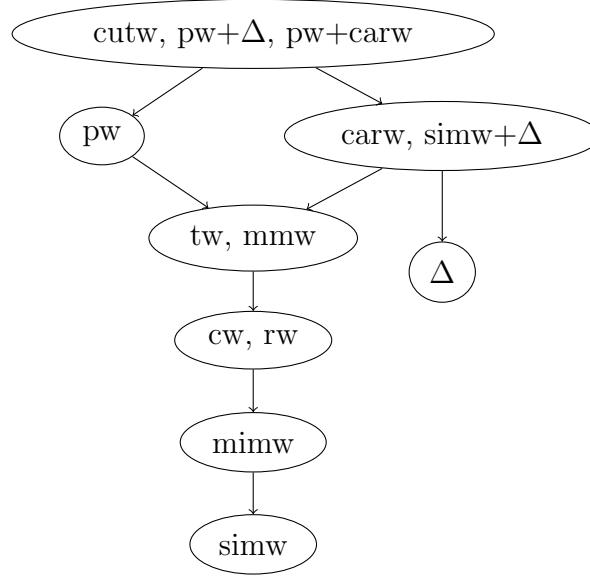


Figura 1.14: Relaciones entre parámetros de anchura. Una flecha del parámetro q al parámetro p indica que $p \preceq q$. Si $p \approx q$ entonces ambos parámetros se grafican dentro del mismo grupo. Las relaciones son transitivas.

Una consecuencia automática del mismo es que las clases de grafos de sim-width y grado acotados son también de carvingwidth acotada. Este resultado es útil para probar que algunas clases de grafos de grado acotado tienen sim-width no acotada, ya que basta con conocer que su carvingwidth (o treewidth, o clique-width, o mim-width) no es acotada. Además, por las desigualdades que vimos antes, las clases de grafos de carvingwidth acotada son también de grado acotado y de sim-width acotada.

Por otro lado, no podemos extender este resultado a otros parámetros conocidos (en concreto, pathwidth y cutwidth) ya que, por ejemplo, los árboles binarios son una clase de grafos de grado a lo sumo 3 y sim-width 1 [124, Proposición 3.1] pero de pathwidth no acotada [155] (y por lo tanto cutwidth no acotada).

En la Figura 1.14 se encuentran resumidas todas estas relaciones, y en la Figura 1.15 encontramos algunas clases de grafos conocidas para las cuales se indica si estos parámetros están acotados por alguna constante. En ambas figuras nos limitamos sólo a los parámetros mencionados en esta tesis.

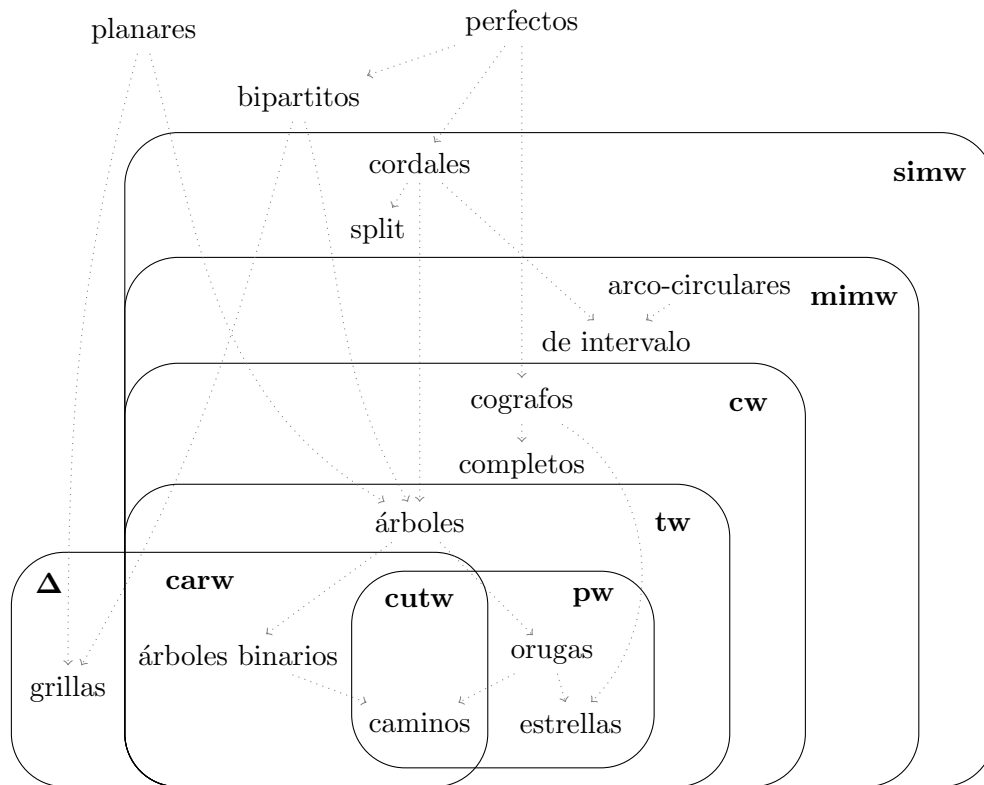


Figura 1.15: Ejemplos de clases de grafos con distintos parámetros acotados por una constante. Se muestran también las inclusiones de las clases de grafos. Los valores exactos de los parámetros en las distintas clases están dados en la Tabla 1.1.

Clase	Δ	cutw	carw	pw	tw	cw	mimw	simw
Árboles binarios	3	∞	3	∞	1	3	1	1
Árboles	∞	∞	∞	∞	1	3	1	1
Caminos	2	1	2	1	1	3	1	1
Orugas	∞	∞	∞	1	1	3	1	1
Estrellas	∞	∞	∞	1	1	2	1	1
Grillas	4	∞	∞	∞	∞	∞	∞	∞
Completos	∞	∞	∞	∞	∞	2	1	1
Cografos	∞	∞	∞	∞	∞	2	≤ 2	≤ 2
De intervalo	∞	∞	∞	∞	∞	∞	1	1
Arco-circulares	∞	∞	∞	∞	∞	∞	2	≤ 2
Split	∞	∞	∞	∞	∞	∞	∞	1
Cordales	∞	∞	∞	∞	∞	∞	∞	1
Bipartitos	∞	∞	∞	∞	∞	∞	∞	∞
Perfectos	∞	∞	∞	∞	∞	∞	∞	∞
Planares	∞	∞	∞	∞	∞	∞	∞	∞

Tabla 1.1: Valores de los parámetros en distintas clases de grafos, donde ∞ indica que dicho parámetro no está acotado en esta clase. Los valores de la columna Δ se desprenden directamente de la definición de la clase o de las inclusiones de clases vistas anteriormente. En [120, Corolario 10] se prueba que una clase particular de grafos, llamados *potencia de hoja* (*leaf power*), tiene mim-width 1, y esta clase incluye a los árboles y a los grafos de intervalo. Para los demás valores, ver [22, 23, 49, 99, 119, 120, 124, 155, 170].

Capítulo 2

Problemas localmente verificables

2.1. Definición

Sean $r \in \mathbb{N}$ y G un grafo. Supongamos que además tenemos lo siguiente:

- un conjunto LABELS y, para cada arista $e \in E(G)$, una *etiqueta* $\ell_e \in \text{LABELS}$;
- un conjunto COLORS y, para cada vértice $v \in V(G)$, un subconjunto no vacío $L_v \subseteq \text{COLORS}$ de posibles *colores* para v ;
- un conjunto de pesos (WEIGHTS, $\preceq, \oplus$) con elemento máximo llamado ERROR;
- una función w (también llamada *función de costo o de peso*) que, dados un vértice $v \in V(G)$ y una asignación de colores $c: N_G^r[v] \rightarrow \text{COLORS}$ tal que $c(u) \in L_u$ para todo $u \in N_G^r[v]$, devuelve un *costo* (o *peso*) en $\text{WEIGHTS} - \{\text{ERROR}\}$; y
- una función *check* (también llamada *función de verificación*) que, dados un vértice $v \in V(G)$ y una asignación de colores $c: N_G^r[v] \rightarrow \text{COLORS}$ tal que $c(u) \in L_u$ para todo $u \in N_G^r[v]$, devuelve TRUE si el vértice v junto con su vecindad de radio r (considerando las etiquetas de las aristas) satisface una cierta condición, y FALSE de lo contrario.

Decimos que una asignación de colores a vértices (también llamada *coloreo*) c es *válida en S* si S es el dominio de c y $c(v) \in L_v$ para todo $v \in S$, y es *propia* si es válida en $V(G)$ y *check* $(v, c|_{N_G^r[v]}) = \text{TRUE}$ para todo $v \in V(G)$. El *costo* (o *peso*) de un coloreo c válido en un conjunto S es $w(c) = \bigoplus_{v \in S} w(v, c|_{N_G^r[v]})$.

Dados todos los elementos previamente definidos, un *problema r -localmente verificable* consiste en encontrar el mínimo costo (de acuerdo al orden $\preceq$) de un coloreo propio. Si tal coloreo no existe, la respuesta debe ser ERROR.

Un *problema r -localmente verificable con propiedades globales* $\Pi = \{P_1, \dots, P_k\}$ es un problema r -localmente verificable donde se requiere adicionalmente que el coloreo propio c dado como salida satisfaga cada una de las propiedades en Π , es decir, que $P_i(c) = \text{TRUE}$ para cada $i \in \llbracket 1, k \rrbracket$. Una propiedad global puede ser, por ejemplo, “el subgrafo inducido por el conjunto $\{v : c(v) = i\}$ es conexo” o “ $|\{v : c(v) = j\}| \leq 1$ ”. En la Sección 2.3 daremos las definiciones de aquellas propiedades que utilizaremos en esta tesis.

2.2. Ejemplos

Para simplificar la escritura, en los ejemplos mencionados a lo largo de esta tesis asumiremos que, a menos que se indique lo contrario, las definiciones de L_v son para todo $v \in V(G)$, y de $w(v, c)$ y $check(v, c)$ para todo $v \in V(G)$ y todo coloreo c válido en $N_G^r[v]$. Además, cuando el problema no emplea etiquetas en las aristas, podemos simplemente definir $LABELS = \{1\}$ y $\ell_e = 1$ para toda $e \in E(G)$, y no escribir su definición. Cuando no se mencione explícitamente a los conjuntos L_v es porque suponemos que $L_v = COLORS$ para todo $v \in V(G)$. Por otro lado, cuando no se mencione explícitamente al conjunto $COLORS$ pero sí a los L_v , supondremos que $COLORS = \bigcup_{v \in V(G)} L_v$.

Comenzaremos con un ejemplo sencillo: MÍNIMO CONJUNTO DOMINANTE modelado como problema 1-localmente verificable. Podemos pensar que tenemos que colorear al grafo con dos colores, digamos s y $\bar{s}$, donde uno de ellos (s) represente a los vértices que pertenecen al conjunto dominante. La función de costo deberá devolver 1 si el vértice recibe el color s y 0 en caso contrario, y la función de verificación deberá devolver **TRUE** si y sólo si existe un vértice en la vecindad cerrada que recibe el color s . Formalmente:

- $COLORS = \{s, \bar{s}\}$,
- $(WEIGHTS, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w(v, c) = \begin{cases} 1 & \text{si } c(v) = s \\ 0 & \text{si no,} \end{cases}$
- $check(v, c) = (\exists u \in N_G[v]. c(u) = s)$.

Notemos que podríamos haber considerado otros colores, por ejemplo 0 y 1, donde 1 indica que el vértice pertenece al conjunto dominante. Así, la definición de la función de costo se puede simplificar como $w(v, c) = c(v)$. Ambos modelos son equivalentes. También podríamos haber definido la función $check$ de la siguiente manera: “ $check(v, c) = \text{TRUE}$ si y sólo si $c(v) = s$ o existe un vecino u de v tal que $c(u) = s$ ”, o de tantas otras maneras. Lo importante no es cómo expresamos su definición, sino que podamos afirmar que la función está bien definida. Lo mismo aplica para w .

Veamos cómo modelar el problema de decidir si un grafo admite un k -coloreo como un problema 1-localmente verificable:

- $COLORS = \llbracket 1, k \rrbracket$,
- $(WEIGHTS, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w(v, c) = 0$,
- $check(v, c) = (\forall u \in N_G(v). c(u) \neq c(v))$.

Como k -COLOREO es un problema de decisión y no de optimización, no utilizamos los costos para optimizar la solución sino precisamente para determinar si una solución es correcta. En efecto, podemos decir que la respuesta es “no” si y sólo si el mínimo costo de un coloreo propio es **ERROR** (que en este caso es 1). Otro posible modelo para este mismo problema es el siguiente:

Problema	WEIGHTS, $\preceq, \oplus$	L_v	$w(v, c)$	$check(v, c)$
k -coloreo	$\{0, 1\}, \leq, \text{máx}$	$\llbracket 1, k \rrbracket$	0	$\forall u \in N_G(v). c(u) \neq c(v)$
Coloreo por listas	$\{0, 1\}, \leq, \text{máx}$	entrada	0	$\forall u \in N_G(v). c(u) \neq c(v)$
H -coloreo	$\{0, 1\}, \leq, \text{máx}$	$V(H)$	0	$\forall u \in N_G(v). c(u) \in N_H(c(v))$
Dominación k -upla	$\mathbb{N}^{+\infty}, \leq, +$	$\{0, 1\}$	$c(v)$	$\sum_{u \in N_G[v]} c(u) \geq k$
Dominación k -upla total	$\mathbb{N}^{+\infty}, \leq, +$	$\{0, 1\}$	$c(v)$	$\sum_{u \in N_G(v)} c(u) \geq k$
k -dominación	$\mathbb{N}^{+\infty}, \leq, +$	$\{0, 1\}$	$c(v)$	$c(v) = 0 \Rightarrow \sum_{u \in N_G(v)} c(u) \geq k$
$\{k\}$ -dominación	$\mathbb{N}^{+\infty}, \leq, +$	$\llbracket 0, k \rrbracket$	$c(v)$	$\sum_{u \in N_G[v]} c(u) \geq k$
Dominación k -arcoíris	$\mathbb{N}^{+\infty}, \leq, +$	$2^{\llbracket 1, k \rrbracket}$	$ c(v) $	$\left \bigcup_{u \in N_G[v]} c(u) \right = k$
Dominación romana	$\mathbb{N}^{+\infty}, \leq, +$	$\llbracket 0, 2 \rrbracket$	$c(v)$	$c(v) = 0 \Rightarrow \exists u \in N_G(v). c(u) = 2$
Conjunto independiente máximo	$\mathbb{N}^{-\infty}, \geq, +$	$\{0, 1\}$	$c(v)$	$c(v) = 1 \Rightarrow \sum_{u \in N_G(v)} c(u) = 0$
Función de $\{k\}$ -empaquetamiento	$\mathbb{N}^{-\infty}, \geq, +$	$\llbracket 0, k \rrbracket$	$c(v)$	$\sum_{u \in N_G[v]} c(u) \leq k$
$\{k\}$ -empaquetamiento limitado	$\mathbb{N}^{-\infty}, \geq, +$	$\{0, 1\}$	$c(v)$	$\sum_{u \in N_G[v]} c(u) \leq k$

Tabla 2.1: Ejemplos de problemas 1-localmente verificables.

- $\text{COLORS} = \llbracket 1, |V(G)| \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, \text{máx})$,
- $w(v, c) = c(v)$,
- $check(v, c) = (\forall u \in N_G(v). c(u) \neq c(v))$.

Aquí, el costo de un coloreo propio es igual al máximo valor entre los colores utilizados, y la solución es aquel coloreo propio que minimiza ese valor. Por lo tanto, para decidir si existe un k -coloreo basta con comparar el costo de la solución con k .

Como vimos, no existe una única forma de modelar un problema dentro de nuestro marco. Algunos modelos pueden tener ventajas algorítmicas sobre otros. Por ejemplo, el primer modelo propuesto para el problema de k -COLOREO utiliza una cantidad constante de colores, lo cual nos permite obtener algoritmos de mejor complejidad que el segundo modelo, donde la cantidad de colores es lineal en la cantidad de vértices. Estudiaremos esto en los Capítulos 3, 4, 5 y 6.

En la Tabla 2.1 mostramos algunos ejemplos de problemas de coloreo, dominación, conjunto independiente y empaquetamiento modelados como problemas 1-localmente verificables. Sus definiciones pueden ser encontradas en el Apéndice A. Para la “versión total” de la mayoría de los problemas de dominación, sólo necesitamos reemplazar $N[v]$ por $N(v)$. Para la “versión pesada” de estos problemas, los costos son parte de la entrada. Para problemas en los que el grafo de entrada es dirigido, podemos considerar etiquetas de aristas que también contengan la información de la dirección de la arista, y las funciones $check$ y w puedan usarlas para distinguir los vecinos de entrada de los de salida.

Entre los problemas r -localmente verificables con $r > 1$, podemos mencionar problemas de coloreo a distancia [48, 54, 55, 82, 97, 101, 104, 159], independencia a distancia [78], y dominación a distancia [108].

2.3. Propiedades globales

A continuación definiremos formalmente las principales *propiedades globales* con las que trabajaremos a lo largo de esta tesis. Sin embargo, cuando presentemos ejemplos de problemas localmente verificables que incluyen propiedades globales, las expresaremos de manera informal, ya que resulta más sencillo de interpretar y sabemos que su formalización es inmediata.

2.3.1. Tamaño

Sean $S \subseteq \text{COLORS}$ un conjunto no vacío de colores y $\sigma_S \subseteq \llbracket 0, |V(G)| \rrbracket$ un conjunto no vacío de enteros no negativos. La propiedad global que exige que la cantidad total de vértices de colores en S sea un elemento de σ_S se llama *propiedad global de tamaño* y se formaliza de la siguiente manera:

$$P(c) = (|\{v \in V(G) : c(v) \in S\}| \in \sigma_S).$$

Por ejemplo, podríamos requerir que la cantidad total de vértices de colores rojo y blanco sea 9, 12 o 18.

En un problema localmente verificable con q colores $\{a_1, \dots, a_q\}$ y grafo de entrada G , podemos también pedir que, para un conjunto $\mathcal{K} \subseteq \text{part}_q(|V(G)|)$ dado, se verifique

$$P(c) = (\exists K \in \mathcal{K}. \forall j \in \llbracket 1, q \rrbracket. |\{v \in V(G) : c(v) = a_j\}| = K_j).$$

A estas también las llamaremos propiedades globales de tamaño. Notemos que de ambas maneras podemos expresar cualquier restricción sobre los tamaños de las clases de colores. Sin embargo, dependiendo del contexto, puede ser más conveniente referirnos a alguna de ellas en particular. Para diferenciar, en el segundo caso diremos concretamente que tenemos “propiedades de tamaño dadas por $\mathcal{K}$ ”.

2.3.2. Conectividad

Sea $C \subseteq \text{COLORS}$ un conjunto no vacío de colores. Una *propiedad global de conectividad positiva* se define como:

$$P(c) = (G[\{v \in V(G) : c(v) \in C\}] \text{ es conexo})$$

mientras que una *propiedad global de conectividad negativa* se define como:

$$P(c) = (G[\{v \in V(G) : c(v) \in C\}] \text{ no es conexo}).$$

En palabras, el subgrafo inducido por los vértices de colores en C debe ser conexo o no-conexo. Por ejemplo, el problema de encontrar un conjunto dominante conexo, definido en [154], se puede modelar de la misma manera que el de conjunto dominante con el agregado de una propiedad global de conectividad positiva $P(c) = (G[\{v \in V(G) : c(v) = s\}] \text{ es conexo})$.

También podemos considerar un conjunto $\mathcal{C}$ de conjuntos no vacíos de colores y pedir que para cada $C \in \mathcal{C}$ se satisfaga una propiedad de conectividad positiva (resp. negativa). En este caso diremos que tenemos “propiedades globales de conectividad positiva (resp. negativa) dadas por $\mathcal{C}$ ”.

2.3.3. Aciclicidad

Sea $A \subseteq \text{COLORS}$ un conjunto no vacío de colores. Una *propiedad global de aciclicidad positiva* se define como:

$$P(c) = (G[\{v \in V(G) : c(v) \in A\}] \text{ es acíclico})$$

y una *propiedad global de aciclicidad negativa* se define como:

$$P(c) = (G[\{v \in V(G) : c(v) \in A\}] \text{ no es acíclico}).$$

En palabras, el subgrafo inducido por los vértices de colores en A debe ser acíclico o no-acíclico. Por ejemplo, el problema COLOREO ACÍCLICO, definido en [105] (ver Apéndice A), se puede modelar de la misma manera que el de coloreo con el agregado de propiedad globales de aciclicidad positiva $P_{a,b}(c) = (G[\{v \in V(G) : c(v) \in \{a, b\}\}] \text{ es acíclico})$ para todo par de colores distintos $a, b \in \text{COLORS}$.

También podemos considerar un conjunto $\mathcal{A}$ de conjuntos no vacíos de colores y pedir que para cada $A \in \mathcal{A}$ se satisfaga una propiedad de aciclicidad positiva (resp. negativa). En este caso diremos que tenemos “propiedades globales de aciclicidad positiva (resp. negativa) dadas por $\mathcal{A}$ ”.

2.3.4. Composición de condiciones locales

Sean $(M, \otimes)$ un monoide conmutativo, X un subconjunto de M y f una función que recibe un vértice v y una asignación de colores válida en $N_G^r[v]$ y devuelve un elemento de M . Una *propiedad global de composición de condiciones locales* se define como:

$$P(c) = \left(\bigotimes_{v \in V(G)} f(v, c|_{N_G^r[v]}) \in X \right).$$

Esta propiedad está inspirada en los problemas LCC de [32]. Aunque en un principio parezca complicada, generaliza varias propiedades relevantes.

En primer lugar, incluye la propiedad global de tamaño, ya que podemos definir $(M, \otimes) = (\mathbb{N}, +)$, $X = \sigma_S$ y la función f de forma que $f(v, c) = 1$ si $c(v) \in S$ y $f(v, c) = 0$ si no. Por este motivo, en algunos capítulos sólo explicaremos cómo resolver esta propiedad global y no la de tamaño. Un punto importante aquí es considerar el tamaño de M , ya que, como veremos más adelante, esto afecta a la complejidad y no cualquier monoide nos dará los mejores resultados. En general, si m es el elemento máximo de σ_S entonces podemos tomar el monoide con $M = \llbracket 0, m+1 \rrbracket$ y $n \otimes n' = \min(n + n', m+1)$, y $X = \sigma_S$. Pero si, por ejemplo, σ_S es el conjunto de números pares entre 0 y $|V(G)|$, podemos tomar $M = \{0, 1\}$ y $n \otimes n' = (n + n' \text{ mód } 2)$, y $X = \{0\}$.

Con esta propiedad también podemos considerar restricciones en la cantidad de aristas que tiene un subgrafo inducido por algún conjunto de colores A . En efecto, podemos definir $f(v, c) = |\{u \in N_G(v) : c(u) \in A\}|$ si $c(v) \in A$ y $f(v, c) = 0$ si no. De esta manera, al sumar todos los $f(v, c|_{N_G^r[v]})$ obtenemos el doble de la cantidad de aristas en el subgrafo inducido por los vértices de colores en A .

Quizás lo más llamativo de esta propiedad es que también generaliza la función de verificación: $(M, \otimes) = (\text{BOOL}, \wedge)$, $X = \{\text{TRUE}\}$ y $f(v, c) = \text{check}(v, c)$. Sin embargo,

con el objetivo de distinguir explícitamente la propiedad local que se debe cumplir para cada vértice, consideramos separadamente a la función de verificación de esta propiedad global. Pero este hecho nos hace ver que las restricciones que tenemos que imponer a f son las mismas que las de *check*, y además que los algoritmos no deben modificarse demasiado para incluir esta propiedad global.

2.4. Restricciones consideradas

Dado que muchos problemas r -localmente verificables son NP-difíciles, es de interés determinar bajo qué condiciones podemos resolverlos eficientemente para una determinada clase de grafos. Es decir, qué tienen que cumplir el conjunto de colores, el conjunto de pesos, las etiquetas en las aristas, las funciones *check* y w , el valor de r y las propiedades globales para que exista un algoritmo de tiempo polinomial que resuelva el problema en esa clase de grafos.

En primer lugar, es importante mencionar que si las funciones propias del problema (*check*, w , $\oplus$, $\min$) no son computables en tiempo polinomial entonces no se puede verificar en tiempo polinomial si una solución es correcta o no (el problema de decisión asociado no es NP), o no se puede calcular o comparar su peso. Por lo tanto, una restricción implícita que consideraremos de ahora en más sobre todos los problemas con los que trabajemos es justamente que estas funciones sean computables en tiempo polinomial. También pediremos que las funciones $\oplus$ y $\min$ sean computables en tiempo constante (si bien esto no es estrictamente necesario, lo pedimos porque facilita los análisis de complejidad y porque los casos donde esto no ocurre son verdaderamente artificiales).

2.4.1. Sistema de vecindades parciales

Definimos un sistema que, informalmente hablando, nos da herramientas para acumular información de los vecinos de un vértice.

Definición 2.4.1. Un *sistema de vecindades parciales* para una instancia de un problema 1-localmente verificable consiste de los siguientes elementos:

- Un conjunto $\mathcal{N}_{v,i}$, para cada vértice $v \in V(G)$ y color $i \in L_v$, junto con una operación binaria cerrada $\boxplus^{v,i}$ sobre $\mathcal{N}_{v,i}$ que es conmutativa y asociativa y tiene un elemento neutro $I_{v,i}$.
- Una función $newN_{v,i}$, para cada $v \in V(G)$ y $i \in L_v$, que dados un vértice $u \in N_G(v)$ y un color $j \in L_u$ devuelve un elemento de $\mathcal{N}_{v,i}$ (posiblemente haciendo uso de la etiqueta de la arista vu).
- Una función $w_{v,i}: \mathcal{N}_{v,i} \rightarrow \text{WEIGHTS}$, para cada $v \in V(G)$ y $i \in L_v$. Esta función debe satisfacer $w_{v,c(v)} \left(\boxplus_{u \in N_G(v)}^{v,c(v)} newN_{v,c(v)}(u, c(u)) \right) = w(v, c)$ para cada vértice $v \in V(G)$ y cada coloreo c válido en $N_G[v]$.
- Una función $check_{v,i}: \mathcal{N}_{v,i} \rightarrow \text{BOOL}$, para cada $v \in V(G)$ y $i \in L_v$. Esta función debe satisfacer $check_{v,c(v)} \left(\boxplus_{u \in N_G(v)}^{v,c(v)} newN_{v,c(v)}(u, c(u)) \right) = check(v, c)$ para cada vértice $v \in V(G)$ y cada coloreo c válido en $N_G[v]$.

En otras palabras, con $newN_{v,i}(u, j)$ creamos nueva información, que dice cómo el hecho de que el vecino u tenga color j afecta a v cuando tiene color i . La operación $\boxplus^{v,i}$ combina dos piezas de información. Para un coloreo c válido en $N_G[v]$, $check_{v,c(v)} \left(\boxplus_{u \in N_G(v)}^{v,c(v)} newN_{v,c(v)}(u, c(u)) \right)$ simplemente verifica una condición sobre toda la información recolectada de los vecinos de v . Finalmente, requerimos la igualdad $check(v, c) = check_{v,c(v)} \left(\boxplus_{u \in N_G(v)}^{v,c(v)} newN_{v,c(v)}(u, c(u)) \right)$ para que el uso de estas herramientas sea análogo al uso de $check(v, c)$. Similar para la función de peso. Nos referimos a los elementos de $\mathcal{N}_{v,i}$ como *vecindades parciales* del vértice v con color i .

Observación 2.4.2. Para cada instancia de un problema 1-localmente verificable existe un sistema de vecindades parciales. A continuación mostraremos cómo construir uno. La idea detrás del siguiente sistema de vecindades parciales es guardar todos los colores asignados a los vértices de v , donde $\perp$ representa que un vecino todavía no fue asignado con ningún color, y $\times$ puede ser pensado como un signo de error. Sean $v \in V(G)$, $i \in L_v$ y supongamos que $N_G(v) = \{u_1, \dots, u_{d_G(v)}\}$. Sea $\mathcal{N}_{v,i}$ el conjunto de todas las $d_G(v)$ -tuplas x tales que $x_h \in L_{u_h} \cup \{\perp, \times\}$ para todo $h \in \llbracket 1, d_G(v) \rrbracket$. Sea $\boxplus^{v,i}$ tal que

$$(n \boxplus^{v,i} n')_h = \begin{cases} n_h & \text{si } n_h = n'_h \text{ o } n'_h = \perp \\ n'_h & \text{si } n_h = \perp \text{ y } n'_h \neq \perp \\ \times & \text{de lo contrario} \end{cases}$$

para todo $h \in \llbracket 1, d_G(v) \rrbracket$. Sea $newN_{v,i}(u_h, j)$ la $d_G(v)$ -tupla que tiene j en su posición h y $\perp$ en todas sus otras posiciones. Sea $x \in \mathcal{N}_{v,i}$. Si $x = (j_1, \dots, j_{d_G(v)})$ con $j_h \in L_{u_h}$ para todo $h \in \llbracket 1, d_G(v) \rrbracket$, sea c el coloreo sobre $N_G[v]$ tal que $c(v) = i$ y $c(u_h) = j_h$ para todo $h \in \llbracket 1, d_G(v) \rrbracket$, y luego definamos $check_{v,i}(x) = check(v, c)$ y $w_{v,i}(x) = w(v, c)$. Si no, sean $check_{v,i}(x) = \text{FALSE}$ y $w_{v,i}(x) = \text{ERROR}$.

Es de suma importancia encontrar sistemas de vecindades parciales que tengan conjuntos $\mathcal{N}_{v,i}$ más pequeños que los descritos anteriormente, ya que esto reduce la complejidad del algoritmo dado en la Sección 3.1. Decimos que un sistema de vecindades parciales es *polinomial* (resp. *constante*) si es tal que $\max\{|\mathcal{N}_{v,i}| : v \in V(G), i \in L_v\}$ es polinomial (resp. constante) en el tamaño de la entrada del problema, y todas las funciones $w_{v,i}$, $check_{v,i}$, eq_n , $\boxplus^{v,i}$ y $newN_{v,i}$ pueden ser computadas en tiempo polinomial (resp. constante) en el tamaño de la entrada. En la Tabla 2.2 mostramos posibles sistemas de vecindades parciales constantes de los problemas 1-localmente verificables listados en la Tabla 2.1. Para simplificar la presentación de algunas complejidades temporales que veremos en los próximos capítulos, diremos que si $\mathcal{P}$ es una cota superior tanto para $\max\{|\mathcal{N}_{v,i}| : v \in V(G), i \in L_v\}$ como para los tiempos de ejecución de todas las funciones $w_{v,i}$, $check_{v,i}$, eq_n , $\boxplus^{v,i}$ y $newN_{v,i}$, entonces el sistema *está acotado por* $\mathcal{P}$ (donde $\mathcal{P}$ no es necesariamente una constante).

Además, decimos que un sistema de vecindades parciales es *universal* si se dan las siguientes propiedades:

- existen un conjunto $\mathcal{N}$ y una operación binaria cerrada $\boxplus$ sobre $\mathcal{N}$ que es conmutativa y asociativa y tiene un elemento neutro I , tales que, para cada vértice v y cada color $i \in L_v$, tenemos que $\mathcal{N}_{v,i} = \mathcal{N}$, la operación $\boxplus^{v,i}$ es la misma que $\boxplus$ y $I_{v,i} = I$;

Problema	$\mathcal{N}_{v,i}$	$n \boxplus^{v,i} n'$	$newN_{v,i}(u, j)$	$w_{v,i}(n)$	$check_{v,i}(n)$
k -coloreo	BOOL	$n \wedge n'$	$j \neq i$	0	n
Coloreo por listas	BOOL	$n \wedge n'$	$j \neq i$	0	n
H -coloreo	BOOL	$n \wedge n'$	$j \in N_H(i)$	0	n
Dominación k -upla	$\llbracket 0, k \rrbracket$	$\min(n + n', k)$	j	i	$n + i \geq k$
Dominación k -upla total	$\llbracket 0, k \rrbracket$	$\min(n + n', k)$	j	i	$n \geq k$
k -dominación	$\llbracket 0, k \rrbracket$	$\min(n + n', k)$	j	i	$i = 0 \Rightarrow n \geq k$
$\{k\}$ -dominación	$\llbracket 0, k \rrbracket$	$\min(n + n', k)$	j	i	$n + i \geq k$
Dominación k -arcoíris	$2^{\llbracket 1, k \rrbracket}$	$n \cup n'$	j	$ i $	$ n \cup i = k$
Dominación romana	BOOL	$n \vee n'$	$j = 2$	i	$i = 0 \Rightarrow n$
Conjunto independiente máximo	$\{0, 1\}$	$\min(n + n', 1)$	j	i	$i = 1 \Rightarrow n = 0$
Función de $\{k\}$ -empaquetamiento	$\llbracket 0, k + 1 \rrbracket$	$\min(n + n', k + 1)$	j	i	$i + n \leq k$
$\{k\}$ -empaquetamiento limitado	$\llbracket 0, k + 1 \rrbracket$	$\min(n + n', k + 1)$	j	i	$i + n \leq k$

Tabla 2.2: Sistemas de vecindades parciales para los problemas de la Tabla 2.1.

- existe una función $newN: V(G) \times \text{COLORS} \rightarrow \mathcal{N}$ tal que, para cada $v \in V(G)$ y cada $i \in L_v$, tenemos que $newN_{v,i}(u, j) = newN(u, j)$ para todo vértice $u \in N_G(v)$ y todo color $j \in L_u$.

En otras palabras, es universal cuando la información a acumular no depende de ningún vértice ni color en particular, todos los vértices acumulan la misma información y luego la utilizan independientemente para sus funciones de verificación y de peso. Este es el caso de los sistemas de la Tabla 2.2, excepto por aquellos de los problemas de las primeras tres filas. No es difícil ver que en realidad estos tres problemas también admiten sistemas de vecindades parciales universales: basta con considerar un conjunto $\mathcal{N} = \text{BOOL}^{|\text{COLORS}|}$ y funciones $\boxplus$ y $newN$ que se comporten como las de la Tabla 2.2 en cada componente $i \in \text{COLORS}$, y finalmente $check_{v,i}(n) = n_i$ (es decir, nos quedamos con la componente correspondiente al color i). Notemos que estos tres sistemas además son constantes si la cantidad de colores es constante, y son polinomiales si la cantidad de colores es logarítmica en el tamaño de la entrada.

2.4.2. Conteo de colores

Sean G un grafo y COLORS un conjunto de colores. Una función f , que recibe un vértice y un coloreo de la vecindad cerrada de dicho vértice, es *contadora de colores* si sólo depende del vértice v , del color que v recibe y, para cada color $a \in \text{COLORS}$, del número de vecinos de v que reciben color a . En otras palabras, f es contadora de colores si existe una función f' tal que

$$f(v, c) = f'(v, c(v), \mu)$$

para cada vértice $v \in V(G)$ y cada coloreo c de la vecindad cerrada de v , donde μ es un vector indexado por los colores de COLORS con $\mu_a = |\{u \in N_G(v) : c(u) = a\}|$

para todo $a \in \text{COLORS}$. A f' la llamaremos *función asociada* a f . Vamos a demandar adicionalmente que el tiempo requerido para computar $f'(v, a, \mu)$ sea polinomial en $|V(G)|$ y además dependa únicamente de μ y de la cantidad de colores. Notar que la definición de funciones contadoras de colores aplica tanto a las funciones de verificación como a las funciones de costo.

Por ejemplo, el modelo para el problema DOMINACIÓN ROMANA de la Tabla 2.1 tiene funciones de peso y verificación que son contadoras de colores, ya que las funciones w' y $check'$, definidas como $w'(v, a, \mu) = a$ y $check'(v, a, \mu) = (a = 0 \Rightarrow \mu_2 \geq 1)$, satisfacen la igualdad pedida. El problema de $\{k\}$ -DOMINACIÓN también cuenta con funciones contadoras de colores: el caso de la función de costo es análogo al anterior, y para la función de verificación podemos considerar $check'(v, a, \mu) = \left(a + \sum_{i=0}^k i \cdot \mu_i \geq k\right)$.

Observación 2.4.3. Todo problema 1-localmente verificable puede reformularse para que sus funciones sean contadoras de colores. En efecto, basta con modificar los colores de modo que el conjunto disponible para cada vértice v sea $\{(a, v) : a \in L_v\}$. Por este motivo es esperable que las restricciones para resolver eficientemente problemas localmente verificables con funciones contadoras de colores también involucren una cota en la cantidad total de colores.

Lema 2.4.4. *Si un problema 1-localmente verificable tiene cantidad constante de colores y además $check$ y w son contadoras de colores, entonces admite un sistema de vecindades parciales universal polinomial.*

Demostración. Sean $check'$ y w' las funciones asociadas a $check$ y w , respectivamente. Basta con definir el siguiente sistema de vecindades parciales universal:

- $\mathcal{N} = \llbracket 0, \Delta(G) \rrbracket^{|\text{COLORS}|}$;
- $(n \boxplus n')_a = \min(n_a + n'_a, \Delta(G))$ para todo $a \in \text{COLORS}$;
- $newN(u, a)_a = 1$ y $newN(u, a)_b = 0$ para todo $b \in \text{COLORS} - \{a\}$;
- $w_{v,a}(n) = w'(v, a, n)$;
- $check_{v,a}(n) = check'(v, a, n)$. □

La noción de funciones contadoras de colores puede extenderse si consideramos vecindades de mayor radio. Sea t un entero positivo. Sean $r_1, \dots, r_t$ enteros positivos tales que $r_1 < \dots < r_t$. Una función f , que recibe un vértice y un coloreo de la vecindad cerrada de radio r_t de dicho vértice, es *contadora de colores a t distancias* $r_1, \dots, r_t$ si sólo depende del vértice v , del color que v recibe y, para cada $a \in \text{COLORS}$ y cada $i \in \llbracket 1, t \rrbracket$, del número de vecinos a distancia a lo sumo r_i de v que reciben color a . En otras palabras, f es contadora de colores a t distancias $r_1, \dots, r_t$ si existe una función f' tal que

$$f(v, c) = f'(v, c(v), \bar{\mu})$$

para cada vértice $v \in V(G)$ y cada coloreo c de la vecindad cerrada de radio r_t de v , donde $\bar{\mu}$ es un vector indexado por COLORS y $\llbracket 1, t \rrbracket$, con $\bar{\mu}_a^i = |\{u \in N_G^{r_i}(v) : c(u) = a\}|$ para todo $a \in \text{COLORS}$ y $i \in \llbracket 1, t \rrbracket$. Cuando $r_1, \dots, r_t$ sean irrelevantes, diremos simplemente que la función es contadora de colores a t distancias.

2.4.3. d -estabilidad

Sea $d \in \mathbb{N}$. Sean G un grafo, COLORS un conjunto de colores y f una función contadora de colores con una función asociada f' . Decimos que f es d -estable si para todo vértice $v \in V(G)$, color $a \in \text{COLORS}$ y vector μ de enteros no negativos indexado por COLORS tenemos $f'(v, a, \mu) = f'(v, a, \mu')$ donde $\mu'_a = \min(d, \mu_a)$ para todo $a \in \text{COLORS}$. En otras palabras, para este tipo de funciones sólo es relevante la cantidad de vecinos de cada color hasta la cota superior d . Notemos que, por lo supuesto para funciones contadoras de colores, el tiempo para computar $f'(v, a, \mu')$ sólo depende de la cantidad de colores y d , por lo tanto cuando ambos valores son constantes entonces este tiempo está acotado por una constante. Cuando una función no es d -estable para ninguna constante d , diremos que es *no-estable*.

Por ejemplo, en la sección anterior vimos que DOMINACIÓN ROMANA cuenta con funciones contadoras de colores, y de acuerdo a esta definición podemos ver fácilmente que además son 1-estables. Para $\{k\}$ -DOMINACIÓN contamos con funciones k -estables, ya que si alguno de los μ_i es mayor a k entonces el resultado de la función es igual que si $\mu_i = k$. De hecho, todos los problemas de la Tabla 2.1 son d -estables para alguna constante d . En el Capítulo 7 mencionamos ejemplos de problemas con funciones no-estables.

Lema 2.4.5. *Si un problema 1-localmente verificable tiene cantidad de colores logarítmica en el tamaño de la entrada y además check y w son d -estables para alguna constante d , entonces admite un sistema de vecindades parciales universal polinomial. Más aún, si la cantidad de colores es constante entonces admite un sistema de vecindades parciales universal constante.*

Demostración. Sean check' y w' las funciones asociadas a check y w , respectivamente. Basta con definir el siguiente sistema de vecindades parciales universal (similar al dado en la demostración del Lema 2.4.4):

- $\mathcal{N} = \llbracket 0, d \rrbracket^{|\text{COLORS}|}$;
- $(n \boxplus n')_a = \min(n_a + n'_a, d)$ para todo $a \in \text{COLORS}$;
- $\text{newN}(u, a)_a = 1$ y $\text{newN}(u, a)_b = 0$ para todo $b \in \text{COLORS} - \{a\}$;
- $w_{v,a}(n) = w'(v, a, n)$;
- $\text{check}_{v,a}(n) = \text{check}'(v, a, n)$. □

Tal y como hicimos en el caso de las funciones contadoras de colores, extenderemos la noción de d -estabilidad a vecindades de mayor radio. Sea t un entero positivo. Sean $r_1, \dots, r_t$ enteros positivos tales que $r_1 < \dots < r_t$. Decimos que una función f contadora de colores a t distancias $r_1, \dots, r_t$ es d -estable a t distancias $r_1, \dots, r_t$ si para todo vértice $v \in V(G)$, color $a \in \text{COLORS}$ y vector $\bar{\mu}$ de enteros no negativos indexado por COLORS y $\llbracket 1, t \rrbracket$ tenemos $f'(v, a, \bar{\mu}) = f'(v, a, \bar{\mu}')$ donde $\bar{\mu}'^i_a = \min(d, \bar{\mu}^i_a)$ para todo $a \in \text{COLORS}$ y todo $i \in \llbracket 1, t \rrbracket$.

2.4.4. m -modularidad

Sea $m \in \mathbb{N}$. Sean G un grafo, COLORS un conjunto de colores y f una función contadora de colores con una función asociada f' . Decimos que f es m -modular si para todo vértice $v \in V(G)$, color $a \in \text{COLORS}$ y vector μ de enteros no negativos indexado por COLORS tenemos $f'(v, a, \mu) = f'(v, a, \mu')$ donde $\mu'_a = (\mu_a \bmod m)$ para todo $a \in \text{COLORS}$. En otras palabras, para este tipo de funciones sólo es relevante la cantidad de vecinos de cada color pero módulo m . En este caso también, por lo supuesto para funciones contadoras de colores, el tiempo para computar $f'(v, a, \mu')$ sólo depende de la cantidad de colores y m , por lo tanto cuando ambos valores son constantes entonces este tiempo está acotado por una constante.

Entre los problemas que cumplen con esta propiedad, podemos mencionar, por ejemplo, al problema de DOMINACIÓN IMPAR, donde se busca un conjunto S de tamaño mínimo tal que $|N_G[v] \cap S|$ sea impar para cada vértice v del grafo G de entrada (a un tal conjunto lo llamaremos *conjunto dominante impar*) [93, 108, 161]. Más general, para todo entero $k \geq 1$ y todo entero $z \in \llbracket 0, k-1 \rrbracket$, el problema de determinar el mínimo tamaño de un conjunto S tal que $|N_G[v] \cap S| \not\equiv z \bmod k$ para todo $v \in V(G)$ (conocido como CONJUNTO DOMINANTE NON- $z \pmod k$) [52]) también cumple con las características descritas arriba.

Lema 2.4.6. *Si un problema 1-localmente verificable tiene cantidad de colores logarítmica en el tamaño de la entrada y además $check$ y w son m -modulares para alguna constante m , entonces admite un sistema de vecindades parciales universal polinomial. Más aún, si la cantidad de colores es constante entonces admite un sistema de vecindades parciales universal constante.*

Demostración. Sean $check'$ y w' las funciones asociadas a $check$ y w , respectivamente. Basta con definir el siguiente sistema de vecindades parciales universal (similar al dado en la demostración del Lema 2.4.4):

- $\mathcal{N} = \llbracket 0, m-1 \rrbracket^{|\text{COLORS}|}$;
- $(n \boxplus n')_a = (n_a + n'_a \bmod m)$ para todo $a \in \text{COLORS}$;
- $newN(u, a)_a = 1$ y $newN(u, a)_b = 0$ para todo $b \in \text{COLORS} - \{a\}$;
- $w_{v,a}(n) = w'(v, a, n)$;
- $check_{v,a}(n) = check'(v, a, n)$. □

También extendemos la noción de m -modularidad a vecindades de mayor radio. Sea t un entero positivo. Sean $r_1, \dots, r_t$ enteros positivos tales que $r_1 < \dots < r_t$. Decimos que una función f contadora de colores a t distancias $r_1, \dots, r_t$ es m -modular a t distancias $r_1, \dots, r_t$ si para todo vértice $v \in V(G)$, color $a \in \text{COLORS}$ y vector $\bar{\mu}$ de enteros no negativos indexado por COLORS y $\llbracket 1, t \rrbracket$ tenemos $f'(v, a, \bar{\mu}) = f'(v, a, \bar{\mu}')$ donde $\bar{\mu}'^i_a = (\bar{\mu}^i_a \bmod m)$ para todo $a \in \text{COLORS}$ y todo $i \in \llbracket 1, t \rrbracket$.

Capítulo 3

Parametrización por treewidth

En este capítulo presentamos un algoritmo XP parametrizado por treewidth para resolver problemas 1-localmente verificables con sistemas de vecindades parciales polinomiales y cantidad polinomial de colores. Este mismo algoritmo es FPT lineal cuando se tienen sistemas de vecindades parciales constantes y cantidad constante de colores. Luego mostramos cómo extenderlo para incluir algunas propiedades globales y estudiaremos qué podemos hacer para resolver problemas r -localmente verificables con $r > 1$. En particular, esto deriva en una parametrización de los problemas r -localmente verificables por carvingwidth. Concluimos analizando qué ocurre con la complejidad de problemas 1-localmente verificables que no satisfacen todas las restricciones impuestas.

3.1. Algoritmo principal

El algoritmo que desarrollamos en esta sección está basado en programación dinámica y procede recorriendo una descomposición en árbol sencilla del grafo de entrada (ver Definición 1.6.7). Concretamente, planteamos una función recursiva que depende, entre otras cosas, de un nodo de la descomposición dada. El comportamiento de la función depende del tipo del nodo (unión, eliminación, inserción u hoja) y utiliza los valores de la función aplicada a los hijos del nodo (si existen). Esta práctica es usual en los algoritmos para resolver algunos problemas localmente verificables previamente estudiados (por ejemplo, en algunas variantes del problema de dominación [165]). En dichos algoritmos se incluyen también ciertos parámetros especiales en la función computada, útiles para acarrear información propia del problema a resolver, normalmente elegidos ad-hoc. En nuestro algoritmo, sin embargo, consideramos una generalización de estos parámetros.

3.1.1. Notación y definiciones

Las siguientes definiciones y notación nos serán de utilidad en el resto de la sección. Para simplificar la notación, si f es una función que dado un x devuelve una función entonces escribiremos f_x en lugar de $f(x)$.

- **Función extendida por un elemento.** Sean $f: X \rightarrow Y$ y $x \notin X$. Luego, la función $f^{x \rightarrow y}: X \cup \{x\} \rightarrow Y \cup \{y\}$ es tal que $f^{x \rightarrow y}(x) = y$ y $f^{x \rightarrow y}(z) = f(z)$ para todo $z \in X$.

- **Función con un elemento menos en su dominio.** Sean $f: X \rightarrow Y$ y $x \in X$. Luego, la función $f^{-x}: X - \{x\} \rightarrow Y$ es tal que $f^{-x}(z) = f(z)$ para todo $z \in X - \{x\}$.
- **Grafo con aristas eliminadas.** Sean H un grafo y S un conjunto de vértices. Luego, H^{-S} es el grafo cuyo conjunto de vértices es $V(H^{-S}) = V(H)$ y cuyo conjunto de aristas es $E(H^{-S}) = E(H) - \{uv : u, v \in S\}$.
- **Asignación válida de vecindades parciales.** Dados un sistema de vecindades parciales, un conjunto de vértices $X \subseteq V(G)$ y una asignación de colores c válida en X , decimos que una *asignación válida de vecindades parciales para c* es una función η de dominio X tal que $\eta(v) \in \mathcal{N}_{v,c(v)}$ para todo $v \in X$.
- **Funciones de peso y verificación para (v, i) .** Dados un vértice $v \in V(G)$ y un color $i \in L_v$, cualquier función $f: \mathcal{N}_{v,i} \rightarrow \text{WEIGHTS}$ se llama *función de peso para (v, i)* y cualquier función $f: \mathcal{N}_{v,i} \rightarrow \text{BOOL}$ se llama *función de verificación para (v, i)* .
- **Función de peso neutro.** Para cada $v \in V(G)$ e $i \in L_v$, sea $w_{v,i}^I: \mathcal{N}_{v,i} \rightarrow \text{BOOL}$ la función tal que $w_{v,i}^I(n) = I_{\oplus}$ para todo $n \in \mathcal{N}_{v,i}$.
- **Función que devuelve funciones de peso neutro.** Dados $X \subseteq V(G)$ y una asignación de colores c válida en X , definimos $\omega^{c,I}$ como la función de dominio X tal que $\omega_v^{c,I}$ es la función $w_{v,c(v)}^I$ para todo $v \in X$.
- **Función de verificación de igualdad.** Para cada $v \in V(G)$, $i \in L_v$ y $n \in \mathcal{N}_{v,i}$, sea $eq_n: \mathcal{N}_{v,i} \rightarrow \text{BOOL}$ la función tal que $eq_n(n') = (n = n')$ para todo $n' \in \mathcal{N}_{v,i}$.
- **Función que devuelve funciones de verificación de igualdad.** Sean $X \subseteq V(G)$, c una asignación de colores válida en X y η una asignación válida de vecindades parciales para c . Luego, definimos $\check{c}^\eta$ como la función de dominio X tal que $\check{c}_v^\eta$ es la función $eq_{\eta(v)}$ para todo $v \in X$.
- **Reducción de una asignación de vecindades parciales.** Sean $X \subseteq V(G)$, c una asignación de colores válida en X , η una asignación válida de vecindades parciales para c y $v \in X$. Luego, la función $\eta^{\sim v}$ de dominio $X - \{v\}$ es tal que $\eta^{\sim v}(u) = \eta(u) \boxplus^{u,c(u)} newN(v, c(v))$ si $u \in X \cap N_G(v)$ y $\eta^{\sim v}(u) = \eta(u)$ si no.
- **Asignación de vecindades parciales neutras.** Sean $X \subseteq V(G)$ y c una asignación de colores válida en X . Luego, la función η_c^I de dominio X es tal que $\eta_c^I(v) = I_{v,c(v)}$ para todo $v \in X$. Observe que η_c^I es una asignación válida de vecindades parciales para c .
- **Vecindad parcial en un subgrafo.** Sean H un subgrafo de G y c una asignación de colores válida en $V(H)$. Definimos NS como la función tal que $NS(v, c, H) = \boxplus_{u \in N_H(v)}^{v,c(v)} newN_{v,c(v)}(u, c(u))$ para todo $v \in V(H)$. Informalmente hablando, $NS(v, c, H)$ es la información que podemos obtener de los vecinos de v en H y la asignación de colores c .

3.1.2. Función a computar

Consideremos una instancia de un problema 1-localmente verificable con un sistema de vecindad parcial asociado. Sea G el grafo de entrada y sea $(T, \{X_t\}_{t \in V(T)})$ una descomposición en árbol sencilla de G .

Para cada G' subgrafo de G , cada $X \subseteq V(G)$ tal que $X \subseteq V(G')$ y $N_G[V(G') - X] \subseteq V(G')$, cada asignación de colores c válida en X , cada η que es una asignación de vecindades parciales válida para c , y cada $\check{c}$ tal que $\check{c}_v \in \{check_{v,c(v)}\} \cup \{eq_n : n \in \mathcal{N}_{v,c(v)}\}$ para todo $v \in X$, decimos que una función f es un $(X, c, \eta, \check{c})$ -coloreo en G' si

- f es una asignación de colores válida en $V(G')$,
- $f(v) = c(v)$ para todo $v \in X$,
- $\check{c}_v(\eta(v) \boxplus^{v,f(v)} \text{NS}(v, f, G')) = \text{TRUE}$ para todo $v \in X$, y
- $check(u, f|_{N_G[u]}) = \text{TRUE}$ para todo $u \in V(G') - X$.

Para un $(X, c, \eta, \check{c})$ -coloreo f en G' y una función ω que dado un vértice $v \in X$ devuelve una función de peso para $(v, f(v))$, definimos el *peso de f bajo ω y η* como $W(f, \omega, \eta) = \left(\bigoplus_{v \in X} \omega_v(\eta(v) \boxplus^{v,f(v)} \text{NS}(v, f, G')) \right) \oplus \left(\bigoplus_{u \in V(G') - X} W(u, f|_{N_G[u]}) \right)$.

Para todo nodo t , sea $\mathcal{D}_t$ el conjunto de todas las tuplas $(S, c, \omega, \eta, \check{c})$ tales que

- $S \subseteq X_t$,
- c es una asignación de colores válida en X_t ,
- ω es una función que, dado un vértice $v \in X_t$, devuelve una función de peso para $(v, c(v))$ tal que $\omega_v \in \{W_{v,c(v)}^I, W_{v,c(v)}\}$,
- η es una asignación de vecindades parciales válida para c , y
- $\check{c}$ es una función que, dado un vértice $v \in X_t$, devuelve una función de verificación para $(v, c(v))$ tal que $\check{c}_v \in \{check_{v,c(v)}\} \cup \{eq_n : n \in \mathcal{N}_{v,c(v)}\}$;

y definamos $\lambda_t: \mathcal{D}_t \rightarrow \text{WEIGHTS}$ como

$$\lambda_t(S, c, \omega, \eta, \check{c}) = \min\{W(f, \omega, \eta) : f \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\}$$

para cada $(S, c, \omega, \eta, \check{c}) \in \mathcal{D}_t$.

Observación 3.1.1. Notar que si no hay $(X_t, c, \eta, \check{c})$ -coloreos en G_t^{-S} entonces tenemos $\lambda_t(S, c, \omega, \eta, \check{c}) = \text{ERROR}$.

El siguiente resultado es inmediato de las definiciones anteriores.

Corolario 3.1.2. Si r es la raíz de T entonces el mínimo costo de un coloreo propio en G es

$$\min\{\lambda_r(\emptyset, c, \omega, \eta_c^I, \check{c}) : (\emptyset, c, \omega, \eta_c^I, \check{c}) \in \mathcal{D}_r, y \\ \omega_v = W_{v,c(v)} \text{ y } \check{c}_v = check_{v,c(v)} \text{ para todo } v \in X_r\}.$$

Mostraremos cómo computar λ de manera recursiva.

Lema 3.1.3 (Nodo hoja). *Sean t un nodo hoja y $X_t = \{v\}$. Luego,*

$$\lambda_t(S, c, \omega, \eta, \check{c}) = \begin{cases} \omega_v(\eta(v)) & \text{si } \check{c}_v(\eta(v)) = \text{TRUE} \\ \text{ERROR} & \text{si no.} \end{cases}$$

Demostración. Notemos que en este caso $V(G_t^{-S}) = \{v\} = X_t$. Analizaremos los dos casos posibles según el valor de $\check{c}_v(\eta(v))$.

Si es FALSE entonces, por definición, no hay $(X_t, c, \eta, \check{c})$ -coloreos en G_t^{-S} . Por lo tanto, $\lambda_t(S, c, \omega, \eta, \check{c}) = \text{ERROR}$, como queríamos probar.

Si por el contrario tenemos $\check{c}_v(\eta(v)) = \text{TRUE}$, entonces, por definición, c es el único $(X_t, c, \eta, \check{c})$ -coloreo posible en G_t^{-S} . Luego, $\lambda_t(S, c, \omega, \eta, \check{c}) = \min\{w(f, \omega, \eta) : f \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\} = w(c, \omega, \eta) = \omega_v(\eta(v))$. $\square$

Lema 3.1.4 (Nodo eliminación). *Sean t un nodo eliminación, s el hijo de t y $X_s - X_t = \{v\}$. Luego,*

$$\lambda_t(S, c, \omega, \eta, \check{c}) = \min_{i \in L_v} \{\lambda_s(S, c^{v \rightarrow i}, \omega^{v \rightarrow w_{v,i}}, \eta^{v \rightarrow I_{v,i}}, \check{c}^{v \rightarrow \text{check}_{v,i}})\}.$$

Demostración. Notemos que $v \notin S$ y también $G_t^{-S} = G_s^{-S}$.

Por definición de λ_s , tenemos que

$$\begin{aligned} & \min_{i \in L_v} \{\lambda_s(S, c^{v \rightarrow i}, \omega^{v \rightarrow w_{v,i}}, \eta^{v \rightarrow I_{v,i}}, \check{c}^{v \rightarrow \text{check}_{v,i}})\} \\ = & \min_{i \in L_v} \{\min\{w(f, \omega^{v \rightarrow w_{v,i}}, \eta^{v \rightarrow I_{v,i}}) : f \text{ es un } (X_s, c^{v \rightarrow i}, \eta^{v \rightarrow I_{v,i}}, \check{c}^{v \rightarrow \text{check}_{v,i}})\text{-coloreo en } G_s^{-S}\}\} \\ = & \min\{w(f, \omega^{v \rightarrow w_{v,i}}, \eta^{v \rightarrow I_{v,i}}) : i \in L_v \wedge f \text{ es un } (X_s, c^{v \rightarrow i}, \eta^{v \rightarrow I_{v,i}}, \check{c}^{v \rightarrow \text{check}_{v,i}})\text{-coloreo en } G_s^{-S}\}. \end{aligned}$$

Sea $i \in L_v$. Afirmamos que toda función f que es un $(X_s, c^{v \rightarrow i}, \eta^{v \rightarrow I_{v,i}}, \check{c}^{v \rightarrow \text{check}_{v,i}})$ -coloreo en G_s^{-S} es también un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} . Además, a la inversa, toda función f que es un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} es también un $(X_s, c^{v \rightarrow f(v)}, \eta^{v \rightarrow I_{v,f(v)}}, \check{c}^{v \rightarrow \text{check}_{v,f(v)}})$ -coloreo en G_s^{-S} . Vamos a probar la primera afirmación (la segunda es similar) mostrando que vale cada uno de los puntos de la definición de $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} :

- $f(w) = c(w)$ para todo $w \in X_t$ (porque es cierto para todo $w \in X_s$),
- f es un coloreo válido en $V(G_s^{-S}) = V(G_t^{-S})$,
- $\text{check}(u, f) = \text{TRUE}$ para todo $u \in V(G_s^{-S}) - X_s = V(G_t^{-S}) - X_t - \{v\}$ y $\text{check}(v, f) = \text{check}_{v,i}(\text{NS}(v, f, G_s^{-S}))$

$$\begin{aligned} &= \text{check}_{v,i}(I_{v,i} \boxplus^{v,i} \text{NS}(v, f, G_s^{-S})) \\ &= \check{c}_v^{v \rightarrow \text{check}_{v,i}}(\eta^{v \rightarrow I_{v,i}}(v) \boxplus^{v,i} \text{NS}(v, f, G_s^{-S})) \\ &= \text{TRUE} \end{aligned}$$

(porque $f(v) = i$), y
- $\check{c}_w(\eta(w) \boxplus^{w,f(w)} \text{NS}(w, f, G_t^{-S})) = \text{TRUE}$ para todo $w \in X_t$ (porque es cierto para todo $w \in X_s$ y $E(G_t^{-S}) = E(G_s^{-S})$).

Es fácil ver que $w(f, \omega^{v \rightarrow w_{v,f(v)}}, \eta^{v \rightarrow I_{v,f(v)}}) = w(f, \omega, \eta)$ para cada f que es un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} . Por lo tanto,

$$\begin{aligned} & \min\{w(f, \omega^{v \rightarrow w_{v,i}}, \eta^{v \rightarrow I_{v,i}}) : i \in L_v \wedge f \text{ es un } (X_s, c^{v \rightarrow i}, \eta^{v \rightarrow I_{v,i}}, \check{c}^{v \rightarrow \text{check}_{v,i}})\text{-coloreo} \\ & \text{en } G_s^{-S}\} \\ &= \min\{w(f, \omega, \eta) : f \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\} \\ &= \lambda_t(S, c, \omega, \eta, \check{c}). \end{aligned} \quad \square$$

Lema 3.1.5 (Nodo inserción). *Sean t un nodo inserción, s el hijo de t y $X_t - X_s = \{v\}$. Sea $n_v = \eta(v) \boxplus^{v,c(v)} \text{NS}(v, c, G_t^{-S}[X_t])$. Luego,*

$$\lambda_t(S, c, \omega, \eta, \check{c}) = \begin{cases} \omega_v(n_v) \oplus \lambda_s(S - \{v\}, c^{-v}, \omega^{-v}, \eta^{\sim v}, \check{c}^{-v}) & \text{si } \check{c}_v(n_v) = \text{TRUE} \\ \text{ERROR} & \text{si no.} \end{cases}$$

Demostración. Observemos que $v \notin V(G_s^{-(S-\{v\})})$ y $N_{G_t^{-S}}[v] \subseteq X_t$ (porque $(T, \{X_i\}_{i \in V(T)})$ es una descomposición en árbol), y que esto implica que $\text{NS}(v, c, G_t^{-S}) = \text{NS}(v, c, G_t^{-S}[X_t])$.

Si $\check{c}_v(n_v) = \text{FALSE}$ entonces, por definición, no existen $(X_t, c, \eta, \check{c})$ -coloreos en G_t^{-S} y también tenemos que $\lambda_t(S, c, \omega, \eta, \check{c}) = \text{ERROR}$.

Ahora supongamos que $\check{c}_v(n_v) = \text{TRUE}$. Se puede probar de manera muy directa que si una función f es un $(X_s, c^{-v}, \eta^{\sim v}, \check{c}^{-v})$ -coloreo en $G_s^{-(S-\{v\})}$ entonces la función $f^{v \rightarrow c(v)}$ es un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} y $w(f^{v \rightarrow c(v)}, \omega, \eta) = \omega_v(n_v) \oplus w(f, \omega^{-v}, \eta^{\sim v})$. Asimismo, también es sencillo probar que para cada $(X_t, c, \eta, \check{c})$ -coloreo g en G_t^{-S} , la función g^{-v} es un $(X_s, c^{-v}, \eta^{\sim v}, \check{c}^{-v})$ -coloreo en $G_s^{-(S-\{v\})}$ y $\omega_v(n_v) \oplus w(g^{-v}, \omega^{-v}, \eta^{\sim v}) = w(g, \omega, \eta)$. Por lo tanto,

$$\begin{aligned} & \lambda_t(S, c, \omega, \eta, \check{c}) \\ &= \min\{w(g, \omega, \eta) : g \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\} \\ &= \min\{\omega_v(n_v) \oplus w(f, \omega^{-v}, \eta^{\sim v}) : f \text{ es un } (X_s, c^{-v}, \eta^{\sim v}, \check{c}^{-v})\text{-coloreo en } G_s^{-(S-\{v\})}\} \\ &= \omega_v(n_v) \oplus \min\{w(f, \omega^{-v}, \eta^{\sim v}) : f \text{ es un } (X_s, c^{-v}, \eta^{\sim v}, \check{c}^{-v})\text{-coloreo en } G_s^{-(S-\{v\})}\} \\ &= \omega_v(n_v) \oplus \lambda_s(S - \{v\}, c^{-v}, \omega^{-v}, \eta^{\sim v}, \check{c}^{-v}). \end{aligned} \quad \square$$

Lema 3.1.6 (Nodo unión). *Sean t un nodo unión, y r y s los hijos de t . Decimos que un par (η_r, η_s) de asignaciones válidas de vecindades parciales para c , es bueno si $\check{c}_v(\eta(v) \boxplus^{v,c(v)} \text{NS}(v, c, G_t^{-S}[X_t]) \boxplus^{v,c(v)} \eta_r(v) \boxplus^{v,c(v)} \eta_s(v)) = \text{TRUE}$ para todo $v \in X_t$. Sea $W(\eta_r, \eta_s) = \bigoplus_{v \in X_t} \omega_v(\eta(v) \boxplus^{v,c(v)} \text{NS}(v, c, G_t^{-S}[X_t]) \boxplus^{v,c(v)} \eta_r(v) \boxplus^{v,c(v)} \eta_s(v))$. Luego,*

$$\lambda_t(S, c, \omega, \eta, \check{c}) = \min_{(\eta_r, \eta_s) \text{ es bueno}} \{W(\eta_r, \eta_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_s})\}.$$

Demostración. Recordemos que $X_t = X_r = X_s$. Para toda asignación de colores f válida en $V(G_t^{-S})$, notamos $f|_r$ (resp. $f|_s$) a la restricción de f a $V(G_r^{-X_r})$ (resp. $V(G_s^{-X_s})$).

Supongamos que existe un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} y sea f uno de ellos. Sean η_r y η_s funciones de dominio X_t tales que $\eta_r(v) = \text{NS}(v, f, G_r^{-X_r})$ y $\eta_s(v) = \text{NS}(v, f, G_s^{-X_s})$ para todo $v \in X_t$. Como $V(G_r^{-X_r}) \cap V(G_s^{-X_s}) = X_t$ entonces $V(G_r^{-X_r}) \cap V(G_s^{-X_s}) \cap N_{G^{-X_t}}(v) = \emptyset$. Más aún, como f es un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} entonces sabemos que, para todo $v \in X_t$, $\text{NS}(v, c, G_t^{-S}[X_t]) \boxplus^{v,c(v)} \eta_r(v) \boxplus^{v,c(v)} \eta_s(v) = \text{NS}(v, f, G_t^{-S})$ y

así $\check{c}_v(\eta(v) \boxplus^{v,c(v)} \text{NS}(v, c, G_t^{-S}[X_t]) \boxplus^{v,c(v)} \eta_r(v) \boxplus^{v,c(v)} \eta_s(v)) = \text{TRUE}$. Por lo tanto, las funciones η_r y η_s forman un par bueno de asignaciones válidas de vecindades parciales para c . Notemos que $w(f, \omega, \eta) = W(\eta_r, \eta_s) \oplus w(f|_r, \omega^{c,I}, \eta_c^I) \oplus w(f|_s, \omega^{c,I}, \eta_c^I)$. Es fácil ver que $f|_r$ es un $(X_r, c, \eta_c^I, \check{c}^{\eta_r})$ -coloreo en $G_r^{-X_r}$ y que $f|_s$ es un $(X_s, c, \eta_c^I, \check{c}^{\eta_s})$ -coloreo en $G_s^{-X_s}$. Por consiguiente,

$$\begin{aligned} w(f, \omega, \eta) &= W(\eta_r, \eta_s) \oplus w(f|_r, \omega^{c,I}, \eta_c^I) \oplus w(f|_s, \omega^{c,I}, \eta_c^I) \\ &\geq W(\eta_r, \eta_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_s}) \\ &\geq \min_{(\eta_r, \eta_s) \text{ es bueno}} \{W(\eta_r, \eta_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_s})\}. \end{aligned}$$

Como $\min\{w(f, \omega, \eta) : f \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\} = \text{ERROR}$ si no hay $(X_t, c, \eta, \check{c})$ -coloreos en G_t^{-S} , obtenemos

$$\begin{aligned} &\lambda_t(S, c, \omega, \eta, \check{c}) \\ &= \min\{w(f, \omega, \eta) : f \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\} \\ &\geq \min_{(\eta_r, \eta_s) \text{ es bueno}} \{W(\eta_r, \eta_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_s})\}. \end{aligned}$$

Para concluir, mostraremos que también vale la otra desigualdad. Si $\min_{(\eta_r, \eta_s) \text{ es bueno}} \{W(\eta_r, \eta_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_s})\} = \text{ERROR}$ entonces el enunciado sigue trivialmente. Si no, el mínimo es alcanzado por un par bueno $(\hat{\eta}_r, \hat{\eta}_s)$, y $\lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\hat{\eta}_r}) \neq \text{ERROR}$ y $\lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\hat{\eta}_s}) \neq \text{ERROR}$. Por tanto existen un $(X_r, c, \eta_c^I, \check{c}^{\hat{\eta}_r})$ -coloreo $\hat{f}_r$ en $G_r^{-X_r}$ y un $(X_s, c, \eta_c^I, \check{c}^{\hat{\eta}_s})$ -coloreo $\hat{f}_s$ en $G_s^{-X_s}$ tales que $w(\hat{f}_r, \omega^{c,I}, \eta_c^I) = \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\hat{\eta}_r})$ y $w(\hat{f}_s, \omega^{c,I}, \eta_c^I) = \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\hat{\eta}_s})$. Sea $\hat{f}$ la función de dominio $V(G_t^{-S})$ tal que $\hat{f}(v) = c(v)$ para todo $v \in X_t$, $\hat{f}(v) = \hat{f}_r(v)$ para todo $v \in V(G_r^{-X_r}) - X_r$, y $\hat{f}(v) = \hat{f}_s(v)$ para todo $v \in V(G_s^{-X_s}) - X_s$. Podemos ver de manera directa que $\hat{f}$ es un $(X_t, c, \eta, \check{c})$ -coloreo en G_t^{-S} , y también que $\hat{f}_r$ (resp. $\hat{f}_s$) es la restricción de $\hat{f}$ a $V(G_r^{-X_r})$ (resp. $V(G_s^{-X_s})$). Luego,

$$\begin{aligned} &\min_{(\eta_r, \eta_s) \text{ es bueno}} \{W(\eta_r, \eta_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\eta_s})\} \\ &= W(\hat{\eta}_r, \hat{\eta}_s) \oplus \lambda_r(X_r, c, \omega^{c,I}, \eta_c^I, \check{c}^{\hat{\eta}_r}) \oplus \lambda_s(X_s, c, \omega^{c,I}, \eta_c^I, \check{c}^{\hat{\eta}_s}) \\ &= W(\hat{\eta}_r, \hat{\eta}_s) \oplus w(\hat{f}_r, \omega^{c,I}, \eta_c^I) \oplus w(\hat{f}_s, \omega^{c,I}, \eta_c^I) \\ &= w(\hat{f}, \omega, \eta) \\ &\geq \min\{w(f, \omega, \eta) : f \text{ es un } (X_t, c, \eta, \check{c})\text{-coloreo en } G_t^{-S}\} \\ &= \lambda_t(S, c, \omega, \eta, \check{c}). \end{aligned} \quad \square$$

Entonces existe un algoritmo sencillo para calcular λ . En efecto, basándonos en la recurrencia dada por los Lemas 3.1.3, 3.1.4, 3.1.5 y 3.1.6, el algoritmo se ejecuta de forma ascendente (“bottom-up” en inglés, esto es, primero por las hojas, luego por los padres de las hojas, y así) calculando $\lambda_t(S, c, \omega, \eta, \check{c}) \in \text{WEIGHTS}$ para cada nodo t y cada $(S, c, \omega, \eta, \check{c}) \in \mathcal{D}_t$. Finalmente, el resultado es obtenido buscando el mínimo entre todos los $\lambda_r(\emptyset, c, \omega, \eta_c^I, \check{c})$ tales que r es la raíz de T , c es una asignación de colores válida en X_r , $\omega_v = w_{v,c(v)}$ para todo $v \in X_r$, y $\check{c}_v = \text{check}_{v,c(v)}$ para todo $v \in X_r$.

3.1.3. Análisis de complejidad

Sean $k = \max\{|X_t| : t \in V(T)\}$, $\mathcal{N} = \max\{|\mathcal{N}_{v,i}| : v \in V(G), i \in L_v\}$ y $\mathcal{C} = \max\{|L_v| : v \in V(G)\}$. Sean $t_{\check{c}}$, t_{ω} , $t_{\boxplus}$, t_{newN} , $t_{\oplus}$, y $t_{\min}$ cotas superiores de los tiempos de ejecución de todas las funciones $check_{v,i}$ y eq_n , $w_{v,i}$ y $w_{v,i}^I$, $\boxplus^{v,i}$, $newN_{v,i}$, $\oplus$, y $\min$, respectivamente. Vamos a suponer que otras operaciones son de tiempo constante.

Recorrer el árbol T requiere tiempo $O(|V(T)|)$. En tiempo $O(k^2|V(T)| + k|E(G)|)$ podemos construir las matrices de adyacencia de todos los grafos $G[X_t]$ con $t \in V(T)$: recorriendo T de arriba hacia abajo (“top-down”) y calculando $N_{G_t}(v) \cap X_t$ sólo para los nodos t que son hijos de un nodo eliminación s con $X_t - X_s = \{v\}$. Además, en tiempo $O(k)$ podemos construir cada una de las extensiones y restricciones de funciones necesarias, y en tiempo $O((t_{\boxplus} + t_{newN})k)$ podemos construir cada una de las η^v necesarias.

Analizaremos separadamente cuatro casos, y en cada uno de ellos la demostración es inmediata.

- Nodo hoja: $O(t_{\check{c}} + t_{\omega})$.
- Nodo eliminación: $O(k^2 + (k + t_{\min})\mathcal{C})$.
- Nodo inserción: $O((t_{\boxplus} + t_{newN})k + t_{\check{c}} + t_{\omega} + t_{\oplus} + k^2)$.
- Nodo unión: $O((t_{\boxplus} + t_{newN})k^2 + t_{\oplus}k + ((t_{\boxplus} + t_{\check{c}} + t_{\omega})k + t_{\oplus} + t_{\min})\mathcal{N}^{2k})$.

Cada uno de ellos se calcula para cada tupla $(S, c, \omega, \eta, \check{c})$ posible. Sabemos que no hay más de $2^k \cdot \mathcal{C}^k \cdot 2^k \cdot \mathcal{N}^k \cdot (1 + \mathcal{N})^k$ de tales tuplas, y que construir cada una de ellas requiere $O(k)$ operaciones.

En síntesis, la complejidad temporal del algoritmo es: $O((t_{\oplus}k + (k + t_{\min})\mathcal{C} + (t_{\boxplus} + t_{newN})k^2 + ((t_{\boxplus} + t_{\check{c}} + t_{\omega})k + t_{\oplus} + t_{\min})\mathcal{N}^{2k})4^k\mathcal{C}^k\mathcal{N}^k(\mathcal{N} + 1)^k|V(T)| + k|E(G)|)$. Por la Proposición 1.6.1, podemos reemplazar $k|E(G)|$ por $k^2|V(G)|$ en la expresión anterior. Si además sabemos que el sistema de vecindades parciales está acotado por algún $\mathcal{P}$, podemos simplificar aún más esta expresión obteniendo una complejidad en $O(k^3 8^k \mathcal{P}^{4k+1} \mathcal{C}^{k+1} |V(T)| + k^2 |V(G)|)$.

A continuación presentamos las conclusiones principales que se pueden obtener a partir del anterior análisis de complejidad y el Corolario 1.6.8.

Teorema 3.1.7. *Si Π es un problema 1-localmente verificable donde cada vértice v tiene $|L_v|$ polinomial (resp. constante) y que admite un sistema de vecindades parciales polinomial (resp. constante), entonces Π es XP (resp. FPT lineal) parametrizado por la treewidth del grafo de entrada.*

Más precisamente, si el sistema de vecindades parciales está acotado por $\mathcal{P}$ entonces existe un algoritmo que resuelve el problema en tiempo $(2\mathcal{P}\mathcal{C})^{O(w)}n$ donde n y w son, respectivamente, la cantidad de vértices y la treewidth del grafo de entrada, y $\mathcal{C} = \max\{|L_v| : v \in V(G)\}$.

En particular, este resultado junto con la Tabla 2.2, implica que todos los problemas listados en la Tabla 2.1 son FPT lineales parametrizados por treewidth. Como vimos en las Secciones 2.4.2, 2.4.3 y 2.4.4, bajo ciertas hipótesis podemos garantizar la existencia de un sistema de vecindades parciales polinomial, obteniendo así el siguiente resultado.

Corolario 3.1.8. *Sea Π un problema 1-localmente verificable en el cual $|\text{COLORS}|$ está acotado por una constante y las funciones de peso y verificación son contadoras de colores. Entonces Π es XP parametrizado por la treewidth del grafo de entrada. Más aún, si las funciones de peso y verificación son d -estables o m -modulares (con d, m constantes), entonces Π es FPT lineal parametrizado por treewidth.*

3.2. Propiedades globales

En esta sección explicaremos cómo modificar el algoritmo anterior con el fin de incluir algunas propiedades globales. La idea general en cada uno de estos casos es modificar la función λ_t extendiéndola con nuevos parámetros $p_1, \dots, p_m$ (es decir, en cada nodo t calcularemos $\lambda_t(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m)$). De esta manera podemos combinar fácilmente varias propiedades globales (no sólo las aquí presentadas, sino cualquier otra que se quiera incorporar en un futuro).

Definiremos recursivamente la noción de $(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m)$ -coloreos en un subgrafo G' del grafo de entrada G . El caso base se da cuando $m = 0$, es decir, cuando no agregamos ningún parámetro y proseguimos como en la sección anterior. Ahora, si $p_1, \dots, p_m$ son parámetros tales que los $(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m)$ -coloreos en G' están bien definidos, entonces, para más parámetros $p_{m+1}, \dots, p_{m+m'}$, definimos un $(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m, p_{m+1}, \dots, p_{m+m'})$ -coloreo en G' como un $(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m)$ -coloreo en G' tal que los parámetros $p_{m+1}, \dots, p_{m+m'}$ satisfacen alguna determinada propiedad (que puede depender también de $S, c, \omega, \eta, \check{c}, p_1, \dots, p_m$). En las siguientes secciones proveeremos los nuevos parámetros asociados con cada propiedad global y daremos la definición de la propiedad que estos nuevos parámetros deben cumplir.

Además, con el mismo espíritu que antes, vamos a notar $\lambda_t(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m)$ al mínimo peso entre todos los $(S, c, \omega, \eta, \check{c}, p_1, \dots, p_m)$ -coloreos en G_t^{-S} . En lo que sigue, escribiremos $\hat{p}$ en lugar de $p_1, \dots, p_m$ para simplificar la escritura. Notemos que $\hat{p}$ es vacío cuando $m = 0$.

3.2.1. Tamaño

Supongamos que queremos que la clase de un color $j \in \text{COLORS}$ tenga un tamaño s , con $s \in \sigma$ para un determinado conjunto σ de enteros no negativos. Consideremos un autómata determinístico de estados finitos $(Q, \{1\}, \delta, q_0, F)$ tal que, para todo $n \in \llbracket 0, |V(G)| \rrbracket$, acepta una cadena de n caracteres 1 consecutivos si y sólo si $n \in \sigma$. Recordemos que, por lo explicado en la Sección 1.2, siempre existe tal autómata y la cantidad de estados podría llegar a ser constante en algunos casos, incluso si el máximo elemento de $\sigma \cap \llbracket 0, |V(G)| \rrbracket$ no es constante. Para un estado $q \in Q$, definimos $\delta^0(q) = q$ y $\delta^n(q) = \delta(\delta^{n-1}(q), 1)$ para todo entero $n \geq 1$.

Sean $state \in Q$ y $accept: Q \rightarrow \text{BOOL}$. Un $(S, c, \omega, \eta, \check{c}, \hat{p}, state, accept)$ -coloreo en G_t^{-S} es un $(S, c, \omega, \eta, \check{c}, \hat{p})$ -coloreo f en G_t^{-S} tal que $accept(\delta^n(state)) = \text{TRUE}$, donde $n = |\{v \in V(G_t^{-S}) : f(v) = j\}|$. Intuitivamente, $state$ guarda el estado del tamaño parcial de la clase de color j y $accept$ verifica que estemos en el estado deseado.

Modificamos el algoritmo de la siguiente manera:

- **Nodo hoja:** Ahora también tenemos que verificar si $accept(state) = \text{TRUE}$.

- **Nodo eliminación:** Para todo $i \in L_v$, sea $state^i = state$ si $i \neq j$ y $state^i = \delta(state, 1)$ si no. Entonces

$$\lambda_t(\dots, state, accept) = \min_{i \in L_v} \{\lambda_s(\dots, state^i, accept)\}.$$

- **Nodo inserción:** Sin cambios (sólo agregando $state$ y $accept$ a λ_s).
 - **Nodo unión:** Para todo $q \in Q$, sea $eq_q: Q \rightarrow \text{BOOL}$ tal que $eq_q(q') = (q = q')$ para todo $q' \in Q$. Luego,
- $$\lambda_t(\dots, state, accept) = \min\{W \oplus \lambda_r(\dots, state, eq_q) \oplus \lambda_s(\dots, q, accept) : q \in Q \wedge \dots\}.$$
- En la raíz r , donde $X_r = \{v\}$, calculamos todos los $\lambda_r(\dots, s_r, a)$, con a tal que $a(s) = (s \in F)$, y $s_r = q_0$ si $c(v) \neq j$ y $s_r = \delta(q_0, 1)$ si no.

Notemos que si quisiéramos restringir el tamaño de $\mathcal{R}$ clases de colores, digamos $j_1, \dots, j_{\mathcal{R}}$, alcanza con asociar a cada clase de color un autómata y los correspondientes parámetros ($state_{j_i}$ y $accept_{j_i}$). Incluso podríamos fácilmente considerar también conjuntos de clases de colores (si quisiéramos, para algún $J \subseteq \text{COLORS}$, que la cantidad de vértices con un color $j \in J$ pertenezca a σ).

La complejidad temporal de este algoritmo modificado depende de la cantidad de estados y del número de clases a restringir. Podemos suponer que verificar si un estado es final es una operación de tiempo constante, y que también lo es calcular $\delta(s, 1)$. Sea $\mathcal{R}$ la cantidad de clases de colores (o conjuntos de clases de colores) a restringir y sea $\mathcal{S}$ el mayor tamaño de los conjuntos de estados de los $\mathcal{R}$ autómatas considerados. Los únicos cambios en la complejidad son:

- **Nodo hoja:** sumar $\mathcal{R}$.
- **Nodo eliminación:** sumar $2\mathcal{R}\mathcal{C}$.
- **Nodo inserción:** sumar $2\mathcal{R}$.
- **Nodo unión:** multiplicar por $\mathcal{S}^{\mathcal{R}}$.
- Cuando multiplicamos por la cantidad de todas las posibles combinaciones de los parámetros de λ_t : agregar un factor $(\mathcal{S}(\mathcal{S} + 1))^{\mathcal{R}}$.

En particular, la complejidad del algoritmo sigue siendo polinomial en $|V(G)|$ si $\mathcal{R}$ está acotado por una constante.

3.2.2. Conectividad positiva

Supongamos que queremos que la clase de un color $j \in \text{COLORS}$ induzca un subgrafo conexo. Agregaremos un nuevo parámetro $comp: X_t \rightarrow \llbracket 1, k \rrbracket$, el cual asignará a los vértices de color j un número natural que representa cuáles de ellos suponemos que están conectados entre sí (los clasifica en supuestas componentes conexas). Formalmente, un $(S, c, \omega, \eta, \check{c}, \hat{p}, comp)$ -coloreo en $G_t^{-\mathcal{S}}$ es un $(S, c, \omega, \eta, \check{c}, \hat{p})$ -coloreo f en $G_t^{-\mathcal{S}}$ tal que si al subgrafo $G_t^{-\mathcal{S}}[f^{-1}(j)]$ le agregamos todas las aristas que conecten pares de vértices $u, v \in X_t \cap f^{-1}(j)$ tales que $comp(u) = comp(v)$ entonces obtenemos un grafo conexo.

En lo que sigue, notaremos X_t^j al conjunto $\{u : u \in X_t \wedge c(u) = j\}$, y $N_t^j(v)$ a $N_G(v) \cap X_t^j$.

- **Nodo hoja:** Sin cambios.
- **Nodo eliminación:** $\lambda_t(\dots, comp) = \min\{\lambda_s(\dots, comp^i) : i \in L_v\}$ donde $comp^i$ es tal que:
 - si $i \neq j$ entonces $comp^i(v) = 1$ y $comp^i(u) = comp(u)$ para todo $u \in X_t^j$, y
 - $comp^j(v) = \begin{cases} \min_{u \in N_t^j(v)} \{comp(u)\} & \text{si } N_t^j(v) \neq \emptyset \\ \text{cualquier valor en } \llbracket 1, k \rrbracket - \{comp(u) : u \in X_t^j\} & \text{si no} \end{cases}$ y, para todo $u \in X_t^j$, $comp^j(u) = comp^j(v)$ si existe $z \in N_t^j(v)$ tal que $comp(u) = comp(z)$, y $comp^j(u) = comp(u)$ si no.

En otras palabras, en el caso que v tenga color j , si v es un vecino de dos o más vértices de color j en diferentes supuestas componentes conexas entonces esas componentes pueden unificarse, y si v no es vecino de ningún otro vértice de color j entonces está en una nueva componente.

- **Nodo inserción:** Si $c(v) \neq j$ entonces el único cambio es agregar $comp^{-v}$ a λ_s . Si no, dividimos el caso correspondiente a $\check{c}_v(n_v) = \text{TRUE}$ en los siguientes casos:

- Si existe $u \in X_s^j$ tal que $comp(v) = comp(u)$ entonces

$$\lambda_t(\dots, comp) = \dots \lambda_s(\dots, comp^{-v}).$$

- Si no existe $u \in X_s^j$ tal que $comp(v) = comp(u)$, y $X_s^j \neq \emptyset$ entonces

$$\lambda_t(\dots, comp) = \text{ERROR}.$$

- Si $X_s^j = \emptyset$ entonces sea $M = (\{q_0, q_1\}, \{1\}, \delta, q_0, \{q_0\})$ un autómata tal que $\delta(q_0, 1) = q_1$ y $\delta(q_1, 1) = q_1$. Usamos M para demandar que la clase de color j sea vacía en G_s , entonces

$$\lambda_t(\dots, comp) = \dots \lambda_s(\dots, q_0, eq_{q_0}).$$

Es decir, si v pertenece a una supuesta componente conexa diferente a la de los demás vértices de color j en X_s , entonces no hay forma de conectar v con ellos, y no hay solución (devolvemos ERROR). Además, si no hay vértices de color j en X_s entonces no puede haber vértices de color j en G_s , ya que $N_G(V(G) - V(G_s)) \cap V(G_s) \subseteq X_s$.

- **Nodo unión:** Para cada $S \subseteq \{comp(v) : v \in X_t^j\}$ sea $comp^S$ una función tal que, para todo $v \in X_t^j$,

$$comp^S(v) = \begin{cases} \min(S) & \text{si } comp(v) \in S \\ comp(v) & \text{si no.} \end{cases}$$

En este nodo necesitamos unificar las diferentes supuestas componentes conexas. Para eso, en una rama unificamos un conjunto S de ellas mientras que en la otra rama unificamos un conjunto R de ellas, de modo que $S \cup R = \{comp(v) : v \in X_t^j\}$ y $|S \cap R| = 1$. Entonces $\lambda_t(\dots, comp) = \min\{\dots \oplus \lambda_r(\dots, comp^S) \oplus \lambda_s(\dots, comp^R) : S \cup R = \{comp(v) : v \in X_t^j\} \wedge |S \cap R| = 1 \wedge \dots\}$.

- En la raíz r , donde $X_r = \{v\}$, la función $comp$ es tal que $comp(v) = 1$.

Al igual que con la propiedad global de tamaño, notemos que podemos agregar tantos parámetros como clases queramos restringir. Sea $\mathcal{J}$ la cantidad de clases (o conjuntos de clases) a restringir. Los únicos cambios en la complejidad son:

- **Nodo hoja:** sumar $\mathcal{J}$.
- **Nodo eliminación:** sumar $(\mathcal{C} - 1 + k^2)\mathcal{J}$.
- **Nodo inserción:** sumar $k\mathcal{J}$.
- **Nodo unión:** multiplicar por $(k2^k)^{\mathcal{J}}$.
- Cuando multiplicamos por la cantidad de todas las posibles combinaciones de los parámetros de λ_t : agregar un factor $(k^k)^{\mathcal{J}}$.

Notemos que como en el nodo inserción tenemos que verificar que la clase de color j sea vacía, necesitamos calcular también $\lambda_t(\dots, q_0, eq_{q_0})$ para todo $t \in T$, pero esto no cambia la complejidad total (por el hecho de que tanto $\mathcal{S}$ como $\mathcal{R}$ están acotados por una constante en este caso).

3.2.3. Aciclicidad positiva

La siguiente solución sólo aplica al caso en el que el grafo de entrada no es dirigido. Este problema puede ser resuelto esencialmente de la misma manera que para la propiedad global de conectividad positiva, donde las únicas diferencias son que en el nodo inserción procedemos como en el algoritmo original y en el nodo eliminación verificamos si v es vecino de al menos dos vértices que pertenecen a la misma supuesta componente conexa (en tal caso encontramos un ciclo, y por lo tanto devolvemos ERROR).

3.2.4. Composición de condiciones locales

Sea $(M, \otimes)$ un monoide conmutativo. Supongamos que queremos satisfacer una expresión de la forma $\bigotimes_{v \in V(G)} f(v, c) \in X$ para algún $X \subseteq M$ y alguna función f que recibe un vértice v y una asignación de colores válida en $N_G[v]$.

Para todo $V \subseteq V(G)$, sea $M(V)$ el conjunto de los diferentes valores que $\bigotimes_{v \in V} f(v, c)$ puede tener. Supongamos que, para cada $V \subseteq V(G)$, $|M(V)|$ está acotado por algún polinomio en el tamaño de la entrada. Supongamos además que calcular $x \otimes y$ y $x \in X$ se puede hacer en tiempo polinomial para todos $x, y \in M$ y $X \subseteq M$. Sea $(M^*, \otimes)$ obtenido al agregar un elemento absorbente E^* a $(M, \otimes)$, y sea I^* el elemento neutro de $(M^*, \otimes)$. Consideremos otros elementos similares al sistema de vecindades parciales, pero para f :

- Un conjunto $\widetilde{\mathcal{N}}_{v,i}$, para cada $v \in V(G)$ y $i \in L_v$, junto con una operación binaria cerrada $\widetilde{\boxplus}^{v,i}$ sobre $\widetilde{\mathcal{N}}_{v,i}$ que es conmutativa y asociativa y tiene un elemento neutro $\widetilde{I}_{v,i}$.
- Una función $\widetilde{newN}_{v,i}$, para cada $v \in V(G)$ y $i \in L_v$, que dados $u \in N_G(v)$ y $j \in L_u$ devuelve un elemento de $\widetilde{\mathcal{N}}_{v,i}$.

- Una función $f_{v,i}: \widetilde{\mathcal{N}}_{v,i} \rightarrow M$, para cada $v \in V(G)$ y $i \in L_v$. Esta función debe cumplir que $f_{v,c(v)} \left(\widetilde{\bigoplus_{u \in N_G(v)}^{v,c(v)} \widetilde{newN}_{v,c(v)}(u, c(u))} \right) = f(v, c)$ para cada vértice $v \in V(G)$ y cada asignación de colores c válida en $N_G[v]$.

En el algoritmo, en cada nodo t , agregamos a λ_t los siguientes parámetros: x , que guarda el estado parcial de la acumulación de valores de f ; $accept$, que verifica si estamos en el estado deseado (similar a la idea de la propiedad global de tamaño); $\tilde{\eta}$, que almacena los valores de la “vecindad parcial para f ” de cada vértice en X_t ; y $\tilde{f}$, que se comporta como $\check{c}$ excepto que en lugar de proveer funciones de codominio BOOL , provee funciones de codominio M^* . Formalmente, un $(S, c, \omega, \eta, \check{c}, \hat{p}, x, accept, \tilde{\eta}, \tilde{f})$ -coloreo g en G_t^{-S} es un $(S, c, \omega, \eta, \check{c}, \hat{p})$ -coloreo en G_t^{-S} tal que $accept(x \otimes n) = \text{TRUE}$, donde $n = \bigotimes_{v \in V(G_t^{-S})} f_{v,g(v)} \left(\tilde{\eta}(v) \widetilde{\bigoplus_{u \in N_{G_t^{-S}}(v)}^{v,c(v)} \widetilde{newN}_{v,g(v)}(u, g(u))} \right)$. Luego procedemos de la siguiente manera.

- **Nodo hoja:** También tenemos que verificar si $accept(x \otimes \tilde{f}_v(\tilde{\eta}(v))) = \text{TRUE}$.
- **Nodo eliminación:**

$$\lambda_t(\dots, x, accept, \tilde{\eta}, \tilde{f}) = \min\{\lambda_s(\dots, x, accept, \tilde{\eta}^{v \rightarrow \tilde{I}_{v,i}}, \tilde{f}^{v \rightarrow f_{v,i}}) : i \in L_v\}.$$

- **Nodo inserción:** Acumulamos en x el valor de $\tilde{f}_v(\tilde{\eta}(v))$, removemos v del dominio de las nuevas funciones, y calculamos las nuevas vecindades parciales para los vértices en X_s .
- **Nodo unión:** Para todo $m \in M^*$, sea $eq_m: M^* \rightarrow \text{BOOL}$ tal que $eq_m(m') = (m = m')$ para todo $m' \in M^*$. Para todo $\tilde{\eta}$, sea $\tilde{f}^{\tilde{\eta}}$ la función definida como $\tilde{f}_v^{\tilde{\eta}}(n) = I^*$ si $n = \tilde{\eta}(v)$ y $\tilde{f}_v^{\tilde{\eta}}(n) = E^*$ si no. Para todo $X \subseteq V(G)$ y asignación de colores c válida en X , sea $\tilde{\eta}_c^{\tilde{I}}$ la función de dominio X tal que $\tilde{\eta}_c^{\tilde{I}}(v) = \tilde{I}_{v,c(v)}$ para todo $v \in X$. Para todo subgrafo H de G y toda asignación de colores c válida en $V(H)$, sea $\widetilde{\text{NS}}(v, c, H) = \widetilde{\bigoplus_{u \in N_H(v)}^{v,c(v)} \widetilde{newN}_{v,c(v)}(u, c(u))}$ para todo $v \in V(H)$. Luego,

$$\lambda_t(\dots, x, accept, \tilde{\eta}, \tilde{f}) = \min \left\{ \dots \oplus \lambda_r(\dots, I^*, eq_{m_r}, \tilde{\eta}_c^{\tilde{I}}, \tilde{f}^{\tilde{\eta}_r}) \oplus \lambda_s(\dots, I^*, eq_{m_s}, \tilde{\eta}_c^{\tilde{I}}, \tilde{f}^{\tilde{\eta}_s}) : \right. \\ m_r \in M(V(G_r) - X_r) \wedge m_s \in M(V(G_s) - X_s) \wedge \\ \left. accept \left(\left(\bigotimes_{v \in X_t} \tilde{f}_v(\tilde{\eta}(v)) \widetilde{\bigoplus_{u \in N_{G_t^{-S}}(v)}^{v,c(v)} \widetilde{newN}_{v,c(v)}(u, c(u))} \right) \otimes x \right) \wedge \dots \right\}.$$

- En la raíz r , donde $X_r = \{v\}$, calculamos todos los $\lambda_r(\dots, I^*, a, \tilde{\eta}_c^{\tilde{I}}, \tilde{f})$, donde a es tal que $a(x) = (x \in X)$, y $\tilde{f}_v = f_{v,c(v)}$.

En consecuencia, la complejidad del algoritmo sigue siendo polinomial.

3.3. El caso $r > 1$

Consideremos el problema de ETIQUETADO $L(2, 1)$ (llamado $L(2, 1)$ -labeling en inglés). En este problema se deben asignar números no-negativos a los vértices de un grafo, minimizando el mayor número asignado, de modo que: para todo par de vecinos, sus números difieran por al menos 2; y para todo par de vértices no adyacentes con un vecino en común, sus números difieran por al menos 1. Como se prueba en [128], una cota superior para el mínimo mayor número asignado es $\Delta(G)^2 + \Delta(G)$ para todo grafo G . Una posible forma de modelar este problema como un problema 2-localmente verificable es la siguiente:

- $\text{COLORS} = \llbracket 0, \Delta(G)^2 + \Delta(G) \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, \text{máx})$,
- $w(v, c) = c(v)$,
- $\text{check}(v, c) = ((\forall u \in N_G(v). |c(u) - c(v)| \geq 2) \wedge (\forall u \in N_G^2(v). |c(u) - c(v)| \geq 1))$.

En [83] se probó que su problema de decisión asociado es NP-completo en grafos de treewidth menor o igual a 2, por lo tanto no es esperable encontrar un algoritmo XP parametrizado por treewidth capaz de resolver todos los problemas r -localmente verificables, incluso cuando nos restringimos a aquellos que tienen funciones 1-estables a 2 distancias 1 y 2.

3.3.1. Parametrización por treewidth más grado máximo, o por carvingwidth

Una forma natural de resolver estos problemas sería tomar la r -ésima potencia del grafo de entrada y allí resolver un problema 1-localmente verificable. El problema con esta idea es que, aunque la treewidth de una clase de grafos $\mathcal{G}$ esté acotada por una constante, la treewidth de la clase $\mathcal{G}^r$ (la clase de las r -ésimas potencias de los grafos en $\mathcal{G}$) puede no estar acotada. Por ejemplo, las estrellas S_n tienen treewidth 1 pero sus cuadrados S_n^2 tienen treewidth n . Sin embargo, el Lema 1.6.9 nos dice que si tanto la treewidth como el grado máximo de $\mathcal{G}$ están acotados entonces la treewidth y el grado de $\mathcal{G}^r$ también están acotados por alguna constante, por lo tanto sí podemos aplicar esta idea.

Recordemos el sistema de vecindades parciales definido en la Observación 2.4.2, en donde $\mathcal{N} \leq (\mathcal{C} + 2)^{\Delta(G)}$. Luego, el siguiente resultado se deduce fácilmente de la complejidad obtenida en la Sección 3.1.3 y el Teorema 1.6.22.

Corolario 3.3.1. *Sea r un entero positivo. Sea Π un problema 1-localmente verificable con cantidad de colores polinomial en el tamaño de la entrada. Luego, Π es XP parametrizado por la carvingwidth del grafo de entrada. Más aún, si $|L_v|$ está acotado por una constante para cada $v \in V(G)$, entonces Π es FPT parametrizado por carvingwidth (e incluso FPT lineal si check y w son computables en tiempo constante).*

Con la ayuda de Lema 1.6.9, extenderemos este resultado a problemas r -localmente verificables. Podemos reducir (en tiempo polinomial) un problema r -localmente verificable a un problema 1-localmente verificable en la r -ésima potencia del grafo de

entrada, donde los conjuntos de pesos y colores permanecen igual que antes, las etiquetas de las aristas ℓ_e ahora además incluyen información sobre la distancia original entre sus extremos (esto se puede modelar, por ejemplo, con una tupla que incluya también el valor anterior, si corresponde), y las funciones *check* y *w* esencialmente iguales teniendo el cuidado de que para conocer $N_G^p(v)$ (los vecinos en el radio p) ahora debemos buscar entre los vértices de $N_{G^r}(v)$ cuáles estaban originalmente a distancia a lo sumo p (y esta información la tenemos porque viene dada en las nuevas etiquetas de las aristas). Luego, a partir del Lema 1.6.9 y el Corolario 3.3.1, sigue este resultado:

Corolario 3.3.2. *Sea r un entero positivo. Sea Π un problema r -localmente verificable con cantidad de colores polinomial en el tamaño de la entrada. Luego, Π es XP parametrizado por la carvingwidth del grafo de entrada. Más aún, si $|L_v|$ está acotado por una constante para cada $v \in V(G)$, entonces Π es FPT parametrizado por carvingwidth (e incluso FPT lineal si *check* y *w* son computables en tiempo constante).*

Veamos ahora que no podemos eliminar la restricción sobre la cantidad de colores. En el problema PLANIFICACIÓN DE EVENTOS PERIÓDICOS (en inglés, *periodic event scheduling problem*, PESP) [135], las instancias consisten de: un grafo dirigido G donde los vértices representan *eventos* y los arcos representan *actividades*, un período de tiempo $T \in \mathbb{N}$, cotas inferiores $\ell_{ij} \in \llbracket 0, T-1 \rrbracket$ y superiores $u_{ij} \in \mathbb{N}$ (con $u_{ij} \geq \ell_{ij}$) para todo arco ij en G . El objetivo es determinar si existe una función $\pi: V(G) \rightarrow \llbracket 0, T-1 \rrbracket$ tal que, para todo arco ij en G , tenemos $\ell_{ij} \leq ((\pi(j) - \pi(i)) \bmod T) \leq u_{ij}$. Este problema es NP-difícil en grafos de carvingwidth 3, pero muy fácilmente podemos modelarlo como un problema 1-localmente verificable: basta con tomar $\text{COLORS} = \llbracket 0, T-1 \rrbracket$ y $\text{check}(i, c) = (\forall j \in N_G^+(v). \ell_{ij} \leq ((\pi(j) - \pi(i)) \bmod T) \leq u_{ij})$. Es más, este modelo admite un sistema de vecindades parciales constante. Como T es parte de la instancia y no necesariamente es polinomial en $|V(G)|$, no podemos extender el Corolario 3.3.2 a cualquier problema r -localmente verificable.

3.3.2. Transformación a problemas 1-localmente verificables

Ciertos problemas r -localmente verificables se pueden modelar también como problemas 1-localmente verificables. A modo de ejemplo, veamos cómo modelar el problema de DOMINACIÓN A DISTANCIA k (ver Apéndice A) como un problema 1-localmente verificable con $k+1$ colores y funciones 1-estables. Para esto, reescribiremos el problema de la siguiente manera: los vértices reciben números enteros entre 0 y k (que representan la distancia a un vértice del conjunto dominante), y los vértices con un número mayor a 0 deben tener al menos un vecino con el número anterior. Luego,

- $\text{COLORS} = \llbracket 0, k \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N} \cup \{+\infty\}, \leq, +)$,
- $w'(v, i, \mu) = \begin{cases} 1 & \text{si } i = 0 \\ 0 & \text{si no,} \end{cases}$
- $\text{check}'(v, i, \mu) = (i > 0 \Rightarrow \mu_{i-1} \geq 1)$.

Notemos que con un argumento similar podemos modelar problemas semejantes que involucren más de dos conjuntos en la partición. La idea es hacer que los

colores indiquen qué tan lejos están los vértices de cada color. Por ejemplo, si tenemos que colorear el grafo con $\{r, g, b\}$ de forma tal que cada vértice esté a distancia a lo sumo 3 de un vértice de color r y a distancia lo sumo 2 de un vértice de color g , siguiendo esta idea para modelar el problema como un problema 1-localmente verificable con funciones 1-estables, nuestro conjunto de colores sería $\{r_{0,1}, r_{0,2}, g_{1,0}, g_{2,0}, g_{3,0}, b_{1,1}, b_{2,1}, b_{3,1}, b_{1,2}, b_{2,2}, b_{3,2}\}$.

Sin embargo, cuando hay restricciones sobre la distancia entre vértices del mismo color, el enfoque anterior no funciona. Explicaremos entonces cómo proceder cuando la distancia requerida en estos casos es 2. Ejemplificaremos con el problema de DOMINACIÓN SEMI-TOTAL, definido en [98] (ver Apéndice A). Primero reescribimos el problema de forma de poder diferenciar dos posibles situaciones para un vértice en D (colores D_1 y D_2) y dos posibles situaciones para un vértice fuera de D (colores $\bar{D}$ y $\bar{D}_*$). Los vértices de color D_1 representan aquellos vértices en D que tienen un vecino en D , los vértices de color D_2 representan aquellos vértices en D que están a distancia 2 de otro vértice en D , los vértices de color $\bar{D}_*$ representan aquellos vértices fuera de D que son el nexo entre dos vértices en D , y los vértices de color $\bar{D}$ representan a todos los restantes vértices. Luego podemos definir:

- $\text{COLORS} = \{D_1, D_2, \bar{D}, \bar{D}_*\},$
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +),$
- $w'(v, a, \mu) = \begin{cases} 1 & \text{si } a = D_1 \text{ o } a = D_2 \\ 0 & \text{si no,} \end{cases}$
- $check'(v, a, \mu) = (a \in \{\bar{D}, D_1\} \Rightarrow \mu_{D_1} + \mu_{D_2} \geq 1) \\ \wedge (a = D_2 \Rightarrow \mu_{\bar{D}_*} \geq 1) \\ \wedge (a = \bar{D}_* \Rightarrow \mu_{D_1} + \mu_{D_2} \geq 2).$

Notemos que en esta reversión de DOMINACIÓN SEMI-TOTAL cambiamos las “responsabilidades” de los vértices: aquellos a cargo de verificar que un vértice de color D_2 esté a distancia 2 de otro vértice en D ahora son los vértices de color $\bar{D}_*$.

3.4. Límites

En esta sección demostramos que no hay esperanzas (excepto que $P = NP$) de encontrar un algoritmo eficiente capaz de resolver todos los problemas localmente verificables que cumplan sólo con un subconjunto de las condiciones impuestas a los problemas en las secciones anteriores.

Ya vimos en la Sección 3.3 que si pudiéramos resolver todos los problemas r -localmente verificables con $r > 1$ entonces podemos modelar el problema de ETIQUETADO $L(2, 1)$, el cual es NP-difícil en grafos de treewidth menor o igual a 2. Otra forma de modelar el problema anterior es como un problema 1-localmente verificable con funciones contadoras de colores (más aún, 1-estables):

- $\text{COLORS} = \llbracket 0, \Delta(G)^2 + \Delta(G) \rrbracket,$
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, \text{máx}),$

- $w'(v, a, \mu) = a$,
- $check'(v, a, \mu) = (\forall i \in \text{COLORS}. \mu_i \neq 0 \Rightarrow (|a - i| \geq 2 \wedge \forall j \in \text{COLORS} - \{i\}. \mu_j \neq 0 \Rightarrow |j - i| \geq 1))$.

Como la cantidad de colores es polinomial, tampoco podemos eliminar la restricción sobre los sistemas de vecindades parciales.

Finalmente, analicemos qué ocurre si la cantidad de colores no es acotada. Consideremos el problema de ASIGNACIÓN DE CANALES (*channel assignment*, en inglés): dado un grafo G en donde cada arista uv tiene asociado un entero positivo $l(uv)$ y dado un entero positivo B , ¿existe una asignación $\varphi: V(G) \rightarrow \llbracket 1, B \rrbracket$ de canales a vértices tal que $|\varphi(u) - \varphi(v)| \geq l(uv)$ para toda arista $uv \in E(G)$? Este problema es NP-completo en grafos de treewidth menor o igual a 3 [140] y es fácil ver que se puede modelar como un problema 1-localmente verificable con un sistema de vecindades parciales constante pero con una cantidad de colores no acotada (ya que B viene dado en la entrada):

- $\text{COLORS} = \llbracket 1, B \rrbracket$,
- $\text{LABELS} = \mathbb{N}^+$ y $\ell_e = l(e)$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w(v, c) = 0$,
- $check(v, c) = (\forall u \in N(v). |c(u) - c(v)| \geq \ell_{uv})$,

y su sistema de vecindades parciales sería:

- $\mathcal{N}_{v,i} = \text{BOOL}$,
- $n \boxplus^{v,i} n' = (n \wedge n')$,
- $newN_{v,i}(u, j) = (|j - i| \geq \ell_{uv})$,
- $w_{v,i}(n) = 0$,
- $check_{v,i}(n) = n$.

Por otro lado, también podemos analizar si es posible obtener un algoritmo general cuya complejidad parametrizada por treewidth sea FPT en lugar de XP. La respuesta a esta cuestión es que no (a menos que $P=NP$), ya que existen problemas 1-localmente verificables, con cantidad polinomial de colores y que admiten un sistema de vecindades parciales polinomial, que son $W[1]$ -difíciles parametrizados por treewidth. Uno de ellos es el de EXTENSIÓN DE PRE-COLOREO (*precoloring extension*, en inglés) [80]: dado un grafo G , un entero k , un subconjunto W de vértices de G y un pre-coloreo $c_W: W \rightarrow \llbracket 1, k \rrbracket$, ¿es posible extender el coloreo a todos los vértices de G usando a lo sumo k colores y de modo que todo par de vecinos reciba distinto color? Expresado como problema 1-localmente verificable:

- $\text{COLORS} = \llbracket 1, k \rrbracket$,
- $L_v = \text{COLORS}$ si $v \notin W$ y $L_v = \{c_W(v)\}$ si $v \in W$,

Restricciones sobre <i>check</i> y <i>w</i>	Restricciones sobre $\max_{v \in V(G)} L_v $		
	Constante	Polinomial	No acotada
Sistema de vecindades parciales constante	FPT lineal	XP, W[1]-h	para-NP-h
Sistema de vecindades parciales polinomial	XP, W[1]-h	XP, W[1]-h	para-NP-h
Computables en tiempo polinomial	?	para-NP-h	para-NP-h

Tabla 3.1: Resumen de los resultados principales sobre la complejidad de problemas 1-localmente verificables parametrizada por treewidth, bajo distintas restricciones sobre la cantidad de colores y las funciones de verificación y peso. Aquí, cuando indicamos para-NP-h (o W[1]-h), estamos diciendo que existe un problema con esas condiciones que es para-NP-h (o W[1]-h) parametrizado por treewidth.

- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w(v, c) = 0$,
- $check(v, c) = (\forall u \in N_G(v). c(u) \neq c(v))$.

Aquí tenemos k colores, donde $k \leq |V(G)|$, y las funciones *check* y *w* admiten un sistema de vecindades parciales constante. Nos resta solamente ver el caso en el que tenemos una cantidad constante de colores y un sistema de vecindades parciales polinomial. Para este caso elegimos el problema de CONJUNTO DOMINANTE DE AFILIACIÓN MÍNIMA (en inglés, *minimum membership dominating set*) definido en [10]: dado un grafo G y un entero k , ¿existe un conjunto dominante S tal que $|N[v] \cap S| \leq k$ para todo $v \in V(G)$? Este problema, el cual es W[1]-difícil parametrizado por pathwidth (y en consecuencia por treewidth) [10], se puede modelar de la siguiente manera como un problema 1-localmente verificable con 2 colores y funciones contadoras de colores (y por lo tanto las funciones admiten un sistema de vecindades parciales universal polinomial, ver Sección 2.4.2):

- $L_v = \{s, \bar{s}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w'(v, i, \mu) = 0$,
- $check'(v, i, \mu) = (1 \leq \mu_s \leq k)$.

En otras palabras, los resultados obtenidos en este capítulo son *ajustados* respecto a las restricciones consideradas. Queda abierta una pregunta respecto al caso con cantidad constante de colores y con funciones de peso y verificación computables en tiempo polinomial pero sin restricciones adicionales, sobre el cual conjeturamos que existe un problema para-NP-difícil con estas condiciones, pero al momento no contamos con ningún ejemplo. Las Tablas 3.1 y 3.2 resumen lo expuesto anteriormente.

Restricciones sobre <i>check</i> y <i>w</i>	Restricciones sobre $\max_{v \in V(G)} L_v $		
	Constante	Polinomial	No acotada
Computables en tiempo polinomial	FPT ⁽¹⁾	XP	para-NP-h

Tabla 3.2: Resumen de los resultados principales sobre la complejidad de problemas r -localmente verificables (para cualquier constante r) parametrizada por carvingwidth (o, equivalentemente, por treewidth más grado), bajo distintas restricciones sobre la cantidad de colores. Aquí, cuando indicamos para-NP-h, estamos diciendo que existe un problema con esas condiciones que es para-NP-h parametrizado por carvingwidth.

(1) FPT lineal si *check* y *w* son computables en tiempo constante.

Capítulo 4

Parametrización por clique-width

En este capítulo probaremos que los problemas 1-localmente verificables que admiten sistemas de vecindades parciales universales polinomiales y tienen cantidad polinomial de colores son XP parametrizados por clique-width. Del mismo análisis se deduce también que los problemas 1-localmente verificables con cantidad acotada de colores y funciones contadoras de colores son XP parametrizados por clique-width, y si además las funciones son d -estables o m -modulares, entonces los problemas son en realidad FPT. Mostraremos cómo incluir propiedades globales de composición de condiciones locales, y por otro lado cómo manejar el caso $r > 1$. Por último, con un análisis similar al de la Sección 3.4, explicaremos por qué no podemos esperar algoritmos mejores.

4.1. Algoritmo

Consideremos un problema 1-localmente verificable Π que admite un sistema de vecindades parciales universal. Llamemos G al grafo de entrada y sea e_G una k -expresión clique-width irredundante de G .

A continuación presentaremos un algoritmo que calcula el peso mínimo de un coloreo propio de G usando la expresión e_G . Para ello, es crucial la siguiente definición.

Definición 4.1.1 ($((C, N)$ -coloreo). Sea e una subexpresión de e_G y sean C y N dos vectores en $\mathcal{N}^k$. Un coloreo c válido en G_e se llama (C, N) -coloreo si satisface las siguientes dos condiciones:

$$(C1) \quad C[i] = \boxplus_{v \in \ell_e^{-1}(i)} \text{new}N(v, c(v)) \text{ para toda etiqueta } i \in \llbracket 1, k \rrbracket;$$

$$(C2) \quad \text{check}_{v, c(v)} \left(N[\ell_e(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \right) = \text{TRUE} \text{ para todo } v \in V(G_e).$$

Definimos el peso de un (C, N) coloreo c de G_e como $w(e, c, N) = \bigoplus_{v \in G_e} w_{v, c(v)} \left(N[\ell_e(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \right)$. El mínimo peso entre todos los posibles (C, N) -coloreos de G_e se nota $\lambda(e, C, N)$, i.e. $\lambda(e, C, N) = \min\{w(e, c, N) : c \text{ es un } (C, N)\text{-coloreo de } G_e\}$. Notemos que si no existe tal coloreo entonces $\lambda(e, C, N) = \text{ERROR}$.

El siguiente lema explica la relación entre los coloreos propios y los (C, N) -coloreos.

Lema 4.1.2. *El peso mínimo de un coloreo propio de G es igual al mínimo entre todos los $\lambda(e_G, C, N_0)$, donde $N_0 \in \mathcal{N}^k$ es la matriz cuyos elementos son todos I (el elemento neutro de $\boxplus$) y $C \in \mathcal{N}^k$ es cualquier matriz tal que $C[i] = I$ para cada $i \in \ell(e_G)$.*

Demostración. Vamos a probar que para cada coloreo propio c de G existe un vector $C \in \mathcal{N}^k$ tal que $C[i] = I$ para todo $i \in \overline{\ell(e_G)}$, y tal que $w(c) \geq \lambda(e_G, C, N_0)$. Por otro lado, vamos a mostrar que para cada vector $C \in \mathcal{N}^k$ tal que $C[i] = I$ para todo $i \in \overline{\ell(e_G)}$, y tal que $\lambda(e_G, C, N_0) \neq \text{ERROR}$, existe un coloreo propio c de G tal que $w(c) = \lambda(e_G, C, N_0)$.

Supongamos que tenemos un coloreo propio c de G . Sea $C \in \mathcal{N}^k$ el vector tal que $C[i] = \bigoplus_{v \in \ell_G^{-1}(i)} \text{newN}(v, c(v))$ para toda etiqueta $i \in \llbracket 1, k \rrbracket$. Claramente, $C[i] = I$ para toda $i \in \overline{\ell(e_G)}$. Además, como c es un coloreo propio de G , para cada vértice $v \in V(G)$ tenemos que $\text{check}_{v, c(v)}(n)$ es verdadero, donde $n = \bigoplus_{u \in N_G(v)} \text{newN}(u, c(u)) = N_0[\ell_e(v)] \boxplus \left(\bigoplus_{u \in N_G(v)} \text{newN}(u, c(u)) \right)$, porque $N_0[\ell_e(v)]$ es igual al neutro de $\boxplus$. Por lo tanto, c es un (C, N_0) -coloreo de G y $w(c) = w(e_G, c, N_0) \geq \lambda(e_G, C, N_0)$.

Ahora supongamos que tenemos una matriz $C \in \mathcal{N}^k$ tal que $C[i] = I$ para toda etiqueta $i \in \overline{\ell(e_G)}$, y tal que $\lambda(e_G, C, N_0) \neq \text{ERROR}$. Sea c un (C, N_0) -coloreo de G de peso $w(e_G, c, N_0)$ mínimo (existe al menos uno ya que $\lambda(e_G, C, N_0) \neq \text{ERROR}$). Demostraremos que c es un coloreo propio de G . Por definición de (C, N_0) -coloreo, c es un coloreo válido, así que sólo resta probar que $\text{check}(v, c|_{N_G[v]})$ es verdadero para cada $v \in V(G)$. Sabemos que para cada $v \in V(G)$ tenemos que $\text{check}_{v, c(v)}(n)$ es verdadero, donde $n = N_0[\ell_e(v)] \boxplus \left(\bigoplus_{u \in N_G(v)} \text{newN}(u, c(u)) \right)$. Como $N_0[\ell_e(v)] = I$ para cada $v \in V(G)$, tenemos que $\text{check}(v, c|_{N_G[v]}) = \text{check}_{v, c(v)} \left(\bigoplus_{u \in N_G(v)} \text{newN}(u, c(u)) \right) = \text{TRUE}$. Más aún, $w(c) = w(e_G, c, N_0) = \lambda(e_G, C, N_0)$. $\square$

En otras palabras, el Lema 4.1.2 nos dice que para resolver Π basta con encontrar el peso mínimo entre todos los (C, N_0) -coloreos del grafo de entrada, donde C y N_0 son como describimos en el lema. El algoritmo que presentaremos se basa precisamente en esta idea: busca el mínimo entre todos los $\lambda(e_G, C, N_0)$, y esto lo logra recorriendo el árbol T_{e_G} de expresión clique-width asociado a e_G de abajo hacia arriba (bottom-up) y calculando recursivamente los valores de cada $\lambda(e, C, N)$, donde e es una subexpresión de e_G y $C, N \in \mathcal{N}^k$. A lo largo de este proceso recursivo, los vectores C y N se comportarán intuitivamente de la siguiente manera: si tenemos un coloreo propio c de G tal que $c|_{V(G_e)}$ es un (C, N) -coloreo de G_e , entonces

- $C[i]$ representa la información que deseamos acumular entre todos los vértices $v \in V(G_e)$ tales que $\ell_e(v) = i$, y
- $N[i]$ representa la información acumulada por los próximos vecinos respecto a e de cada vértice v con $\ell_e(v) = i$.

Describiremos recursivamente el cálculo de $\lambda(e, C, N)$ distinguiendo en cuatro casos según el tipo de operación clique-width en la raíz de T_e . Para simplificar la lectura, en los siguientes cuatro lemas supondremos que es están dados los vectores C y N en $\mathcal{N}^k$.

Lema 4.1.3 (Creación de un nuevo vértice: $i(v)$). *Si $C[j] = I$ para toda etiqueta $j \in \llbracket 1, k \rrbracket - \{i\}$, entonces*

$$\begin{aligned} \lambda(i(v), C, N) &= \min\{w_{v,a}(N[i]) : \\ &\quad (a) \ a \in L_v \\ &\quad (b) \ C[i] = \text{newN}(v, a) \end{aligned}$$

$$(c) \text{ check}_{v,a}(N[i]) = \text{TRUE}\}.$$

Si no, $\lambda(i(v), C, N) = \text{ERROR}$.

Demostración. Primero notemos que $G_{i(v)}$ es el grafo que consiste de un único vértice v con etiqueta i . Por lo tanto, si $C[j] \neq I$ para alguna etiqueta $j \in \llbracket 1, k \rrbracket - \{i\}$ entonces no existe ningún (C, N) -coloreo de $G_{i(v)}$ y $\lambda(i(v), C, N) = \text{ERROR}$. Supongamos entonces que $C[j] = I$ para toda $j \in \llbracket 1, k \rrbracket - \{i\}$.

A continuación reescribiremos la definición de (C, N) -coloreo, adaptándola específicamente a coloreos de $G_{i(v)}$. Un coloreo $c : \{v\} \rightarrow L_v$ es un (C, N) -coloreo de $G_{i(v)}$ si satisface las siguientes dos condiciones:

- (C1) $C[i] = \text{newN}(v, c(v))$, ya que $\ell_{i(v)}^{-1}(j) = \emptyset$ para toda $j \in \llbracket 1, k \rrbracket - \{i\}$ y además $\bigoplus_{u \in \ell_{i(v)}^{-1}(i)} \text{newN}(u, c(u)) = \text{newN}(v, c(v))$ porque $\ell_{i(v)}^{-1}(i) = \{v\}$;
- (C2) $\text{check}_{v,c(v)}(N[i]) = \text{TRUE}$, ya que, como v no tiene vecinos en $G_{i(v)}$, tenemos que $\text{check}_{v,c(v)}\left(N[i] \boxplus \left(\bigoplus_{u \in N_{G_{i(v)}}(v)} \text{newN}(u, c(u))\right)\right) = \text{check}_{v,c(v)}(N[i])$.

Es decir, $c : \{v\} \rightarrow L_v$ es un (C, N) -coloreo de $G_{i(v)}$ si y sólo si existe $a \in L_v$ tal que $a = c(v)$, $C[i] = \text{newN}(v, a)$ y $\text{check}_{v,a}(N[i]) = \text{TRUE}$. Por lo tanto, se cumple la igualdad del enunciado. $\square$

Lema 4.1.4 (Unión disjunta: $e_1 \oplus e_2$). Sean N_1 y N_2 dos vectores en $\mathcal{N}^k$ tales que:

- $N_1[i] = N[i]$ para toda etiqueta $i \in \ell(e_1)$,
- $N_1[i] = I$ para toda etiqueta $i \in \overline{\ell(e_1)}$, y
- N_2 está definido análogamente respecto a e_2 .

Luego,

$$\begin{aligned} \lambda(e_1 \oplus e_2, C, N) &= \min\{\lambda(e_1, C_1, N_1) \oplus \lambda(e_2, C_2, N_2) : \\ &\quad (a) C_1, C_2 \in \mathcal{N}^k \\ &\quad (b) C_1[i] = I \text{ para toda } i \in \overline{\ell(e_1)} \\ &\quad (c) C_2[i] = I \text{ para toda } i \in \overline{\ell(e_2)} \\ &\quad (d) C[i] = C_1[i] \boxplus C_2[i] \text{ para toda } i \in \llbracket 1, k \rrbracket\}. \end{aligned}$$

Demostración. Para facilitar la lectura, llamemos α a $\min\{\lambda(e_1, C_1, N_1) \oplus \lambda(e_2, C_2, N_2) : \text{se cumplen (a), (b), (c), (d)}\}$.

Primero demostraremos que $\lambda(e_1 \oplus e_2, C, N) \geq \alpha$. Si $\lambda(e_1 \oplus e_2, C, N) = \text{ERROR}$ entonces estamos. Así que supongamos que $\lambda(e_1 \oplus e_2, C, N) \neq \text{ERROR}$ y sea c un (C, N) -coloreo de $G_{e_1 \oplus e_2}$ cuyo peso $w(e_1 \oplus e_2, c, N)$ sea igual a $\lambda(e_1 \oplus e_2, C, N)$. Necesitamos probar que existen C_1 y C_2 en $\mathcal{N}^k$ que satisfacen (b), (c), (d) y tales que el peso de c es al menos $\lambda(e_1, C_1, N_1) \oplus \lambda(e_2, C_2, N_2)$. Sean $c_1 = c|_{V(G_{e_1})}$ y $c_2 = c|_{V(G_{e_2})}$. Luego, definimos $C_1[i] = \bigoplus_{v \in \ell_{e_1}^{-1}(i)} \text{newN}(v, c_1(v))$ para cada etiqueta $i \in \llbracket 1, k \rrbracket$, y definimos C_2 análogamente respecto a e_2 y c_2 . En consecuencia, las condiciones (b) y (c) se satisfacen: si $i \in \overline{\ell(e_1)}$ entonces $\ell_{e_1}^{-1}(i) = \emptyset$, así $C_1[i] = I$, y análogamente para C_2 . La condición (d) también es satisfecha porque c es un (C, N) -coloreo de

$G_{e_1 \oplus e_2}$, $V(G_{e_1})$ y $V(G_{e_2})$ son disjuntos, $\ell_{e_1}(v) = \ell_{e_1 \oplus e_2}(v)$ para cada $v \in V(G_{e_1})$, $\ell_{e_2}(v) = \ell_{e_1 \oplus e_2}(v)$ para cada $v \in V(G_{e_2})$, y $\boxplus$ es asociativa y conmutativa. Ahora mostraremos que c_1 es un (C_1, N_1) -coloreo de G_{e_1} . La condición (C1) es satisfecha trivialmente por definición de C_1 . Para mostrar que la condición (C2) es satisfecha, vamos a probar que $check_{v, c_1(v)}(n')$ es verdadero para cada vértice $v \in V(G_{e_1})$, donde $n' = N_1[\ell_{e_1}(v)] \boxplus \left(\boxplus_{u \in N_{G_{e_1}}(v)} newN(u, c_1(u)) \right)$. Como c es un (C, N) -coloreo de $G_{e_1 \oplus e_2}$, tenemos que $check_{v, c(v)}(n)$ es verdadero para cada $v \in V(G_{e_1 \oplus e_2})$, donde $n = N[\ell_{e_1 \oplus e_2}(v)] \boxplus \left(\boxplus_{u \in N_{G_{e_1 \oplus e_2}}(v)} newN(u, c(u)) \right)$. Por definición de N_1 y como $\ell_{e_1}(v) = \ell_{e_1 \oplus e_2}(v)$ para cada $v \in V(G_{e_1})$, tenemos $N_1[\ell_{e_1}(v)] = N[\ell_{e_1 \oplus e_2}(v)]$ para cada $v \in V(G_{e_1})$. Además, $N_{G_{e_1}}(v) = N_{G_{e_1 \oplus e_2}}(v)$ para cada $v \in V(G_{e_1})$ y $c_1(u) = c(u)$ para cada $u \in V(G_{e_1})$. Por lo tanto, $n' = n$ y así $check_{v, c_1(v)}(n')$ es verdadero para cada $v \in V(G_{e_1})$. Usando argumentos similares, podemos probar que c_2 es un (C_2, N_2) -coloreo de G_{e_2} . Más aún, $w(e_1 \oplus e_2, c, N) = w(e_1, c_1, N_1) \oplus w(e_2, c_2, N_2)$. En consecuencia, $\lambda(e_1 \oplus e_2, C, N) = w(e_1 \oplus e_2, c, N) = w(e_1, c_1, N_1) \oplus w(e_2, c_2, N_2) \geq \lambda(e_1, C_1, N_1) \oplus \lambda(e_2, C_2, N_2) \geq \alpha$.

Ahora demostraremos que $\lambda(e_1 \oplus e_2, C, N) \leq \alpha$. Consideremos dos vectores C_1 y C_2 en $\mathcal{N}^k$ que satisfacen (b), (c), (d). Si $\lambda(e_1, C_1, N_1) = \text{ERROR}$ o $\lambda(e_2, C_2, N_2) = \text{ERROR}$, entonces estamos. Supongamos entonces que esto no ocurre. Vamos a construir un (C, N) -coloreo c de $G_{e_1 \oplus e_2}$ tal que $w(e_1 \oplus e_2, c, N) = \lambda(e_1, C_1, N_1) \oplus \lambda(e_2, C_2, N_2)$. Sean c_1 un (C_1, N_1) -coloreo de G_{e_1} cuyo peso es $\lambda(e_1, C_1, N_1)$ y c_2 un (C_2, N_2) -coloreo de G_{e_2} cuyo peso es $\lambda(e_2, C_2, N_2)$. Sea $c = c_1 \cup c_2$ (notemos que c está bien definido porque $V(G_{e_1})$ y $V(G_{e_2})$ son conjuntos disjuntos). Claramente, c es un coloreo válido. Ahora mostraremos que c es un (C, N) -coloreo de $G_{e_1 \oplus e_2}$. Comenzaremos con la condición (C1). Consideremos $i \in \llbracket 1, k \rrbracket$. Como c_1 es un (C_1, N_1) -coloreo de G_{e_1} y c_2 es un (C_2, N_2) -coloreo de G_{e_2} , tenemos $C_1[i] = \boxplus_{v \in \ell_{e_1}^{-1}(i)} newN(v, c_1(v))$ y $C_2[i] = \boxplus_{v \in \ell_{e_2}^{-1}(i)} newN(v, c_2(v))$. Luego,

$$\begin{aligned}
C[i] &= C_1[i] \boxplus C_2[i] \\
&= \left(\boxplus_{v \in \ell_{e_1}^{-1}(i)} newN(v, c_1(v)) \right) \boxplus \left(\boxplus_{v \in \ell_{e_2}^{-1}(i)} newN(v, c_2(v)) \right) \\
&= \left(\boxplus_{v \in \ell_{e_1}^{-1}(i)} newN(v, c(v)) \right) \boxplus \left(\boxplus_{v \in \ell_{e_2}^{-1}(i)} newN(v, c(v)) \right) \\
&= \boxplus_{v \in \ell_{e_1}^{-1}(i) \cup \ell_{e_2}^{-1}(i)} newN(v, c(v)) \\
&= \boxplus_{v \in \ell_e^{-1}(i)} newN(v, c(v)).
\end{aligned}$$

La última igualdad se debe simplemente a que $V(G_{e_1 \oplus e_2}) = V(G_{e_1}) \cup V(G_{e_2})$, a que para cada $v \in V(G_{e_1})$ vale que $\ell_{e_1}(v) = \ell_{e_1 \oplus e_2}(v)$ y a que para cada $v \in V(G_{e_2})$ vale que $\ell_{e_2}(v) = \ell_{e_1 \oplus e_2}(v)$. Ahora nos enfocaremos en la condición (C2). Consideremos $v \in V(G_{e_1 \oplus e_2})$. Supongamos sin pérdida de generalidad que $v \in V(G_{e_1})$. Vamos a probar que $check_{v, c(v)}(n)$ es verdadero, donde $n = N[\ell_{e_1 \oplus e_2}(v)] \boxplus \left(\boxplus_{u \in N_{G_{e_1 \oplus e_2}}(v)} newN(u, c(u)) \right)$. Como c_1 es un (C_1, N_1) -coloreo de G_{e_1} , sabemos que $check_{v, c_1(v)}(n')$ es verdadero, donde $n' = N_1[\ell_{e_1}(v)] \boxplus \left(\boxplus_{u \in N_{G_{e_1}}(v)} newN(u, c_1(u)) \right)$. Usando argumentos similares a los anteriores, obtenemos que $n' = n$. Como además $c(v) = c_1(v)$, concluimos que $check_{v, c(v)}(n)$ es verdadero para cada $v \in V(G)$, y así la condición (C2) es satisfecha. Luego, c es un (C, N) -coloreo de $G_{e_1 \oplus e_2}$. Más aún, $w(e_1 \oplus e_2, c, N) = w(e_1, c_1, N_1) \oplus w(e_2, c_2, N_2)$. $\square$

Lema 4.1.5 (Unión completa: $\eta_{i,j}(e)$). Sea $N_e \in \mathcal{N}^k$ tal que

- $N_e[i] = N[i] \boxplus C[j]$;
- $N_e[j] = N[j] \boxplus C[i]$;
- $N_e[h] = N[h]$ para cada $h \in \llbracket 1, k \rrbracket - \{i, j\}$.

Luego, $\lambda(\eta_{i,j}(e), C, N) = \lambda(e, C, N_e)$.

Demostración. En primer lugar, como las etiquetas de los vértices no cambian entre $G_{\eta_{i,j}(e)}$ y G_e , usaremos simplemente ℓ para referirnos a ambos etiquetados ℓ_e y $\ell_{\eta_{i,j}(e)}$. Asimismo, como $V(G_{\eta_{i,j}(e)}) = V(G_e)$, llamaremos V a este conjunto.

Sea $c: V \rightarrow \text{COLORS}$ un coloreo válido. Vamos a demostrar que c es un (C, N) -coloreo de $G_{\eta_{i,j}(e)}$ si y sólo si c es un (C, N_e) -coloreo de G_e . La condición (C1) se cumple trivialmente en ambas implicaciones (porque es exactamente la misma). Luego, basta con probar que, suponiendo (C1), tenemos $N_e[\ell(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) = N[\ell(v)] \boxplus \left(\boxplus_{u \in N_{G_{\eta_{i,j}(e)}}(v)} \text{new}N(u, c(u)) \right)$ para cada vértice $v \in V$.

Si $\ell(v) \neq i, j$ entonces $N_e[\ell(v)] = N[\ell(v)]$ por definición de N_e y claramente $N_{G_e}(v) = N_{G_{\eta_{i,j}(e)}}(v)$. Por otro lado, si $\ell(v) = i$ (si $\ell(v) = j$ la prueba es análoga) entonces

$$\begin{aligned} & N_e[i] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \\ &= N[i] \boxplus C[j] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \\ &= N[i] \boxplus \left(\boxplus_{u \in \ell^{-1}(j)} \text{new}N(u, c(u)) \right) \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \\ &= N[i] \boxplus \left(\boxplus_{u \in \ell^{-1}(j) \cup N_{G_e}(v)} \text{new}N(u, c(u)) \right) \\ &= N[i] \boxplus \left(\boxplus_{u \in N_{G_{\eta_{i,j}(e)}}(v)} \text{new}N(u, c(u)) \right). \end{aligned}$$

Las últimas dos igualdades siguen por el hecho de que cada vértice con etiqueta j es un vecino de v en $G_{\eta_{i,j}(e)}$ pero es un no-vecino de v en G_e (recordemos que estamos trabajando con expresiones clique-width irredundantes).

Finalmente, por la relación entre los vectores N y N_e que acabamos de mostrar, tenemos que $w(\eta_{i,j}(e), c, N) = w(e, c, N_e)$ y entonces $\lambda(\eta_{i,j}(e), C, N) = \lambda(e, C, N_e)$. $\square$

Lema 4.1.6 (Renombre de etiqueta: $\rho_{i \rightarrow j}(e)$). Sea $N_e \in \mathcal{N}^k$ tal que

- $N_e[i] = N[j]$, y
- $N_e[h] = N[h]$ para cada $h \in \llbracket 1, k \rrbracket - \{i\}$.

Si $C[i] = I$, entonces

$$\begin{aligned} \lambda(\rho_{i \rightarrow j}(e), C, N) &= \min\{\lambda(e, C_e, N_e) : \\ &\quad (a) C_e \in \mathcal{N}^k \\ &\quad (b) C[j] = C_e[i] \boxplus C_e[j] \\ &\quad (c) C_e[h] = C[h] \text{ para toda } h \in \llbracket 1, k \rrbracket - \{i, j\}\}. \end{aligned}$$

Si no, $\lambda(\rho_{i \rightarrow j}(e), C, N) = \text{ERROR}$.

Demostración. Si $C[i] \neq I$, entonces, por definición, no existe ningún (C, N) -coloreo de $G_{\rho_{i \rightarrow j}(e)}$ y $\lambda(\rho_{i \rightarrow j}(e), C, N) = \text{ERROR}$. Por lo tanto podemos suponer que $C[i] = I$. Llamemos α a $\min\{\lambda(e, C_e, N_e) : (a), (b), (c) \text{ son satisfechas}\}$.

Primero probaremos que $\lambda(\rho_{i \rightarrow j}(e), C, N) \geq \alpha$. Sea c un (C, N) -coloreo de $G_{\rho_{i \rightarrow j}(e)}$ cuyo peso es igual a $\lambda(\rho_{i \rightarrow j}(e), C, N)$. Vamos a probar que existe un vector $C_e \in \mathcal{N}^k$ que satisface (b), (c) y es tal que c es un (C_e, N_e) -coloreo de G_e . Definimos $C_e[h] = \bigsqcup_{v \in \ell_e^{-1}(h)} \text{new}N(v, c(v))$ para cada $h \in \llbracket 1, k \rrbracket$. Como c es un (C, N) -coloreo de $G_{\rho_{i \rightarrow j}(e)}$, entonces $C[h] = \bigsqcup_{v \in \ell_{\rho_{i \rightarrow j}(e)}^{-1}(h)} \text{new}N(v, c(v))$ para cada $h \in \llbracket 1, k \rrbracket$. Por lo tanto, (c) es trivialmente satisfecha, ya que $\ell_{\rho_{i \rightarrow j}(e)}(v) = \ell_e(v)$ para todo vértice v cuya etiqueta no es ni i ni j en G_e . Más aún, se cumplen las siguientes igualdades:

$$\begin{aligned} C_e[i] \sqcup C_e[j] &= \left(\bigsqcup_{v \in \ell_e^{-1}(i)} \text{new}N(v, c(v)) \right) \sqcup \left(\bigsqcup_{v \in \ell_e^{-1}(j)} \text{new}N(v, c(v)) \right) \\ &= \bigsqcup_{v \in \ell_e^{-1}(i) \cup \ell_e^{-1}(j)} \text{new}N(v, c(v)) \\ &= \bigsqcup_{v \in \ell_{\rho_{i \rightarrow j}(e)}^{-1}(j)} \text{new}N(v, c(v)) \\ &= C[j] \end{aligned}$$

y así, la condición (b) es satisfecha también. Veamos ahora que c es un (C_e, N_e) -coloreo de G_e . Sabemos que c es un coloreo válido porque c es un (C, N) -coloreo de $G_{\rho_{i \rightarrow j}(e)}$. También, (C1) es trivialmente satisfecha por definición de C_e . En cuanto a (C2), tenemos que probar que $\text{check}_{v, c(v)}(n')$ es verdadero para cada v en G_e , donde $n' = N_e[\ell_e(v)] \sqcup \left(\bigsqcup_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right)$. Como c es un (C, N) -coloreo de $G_{\rho_{i \rightarrow j}(e)}$, sabemos que $\text{check}_{v, c(v)}(n)$ es verdadero para cada $v \in V(G_{\rho_{i \rightarrow j}(e)})$, donde $n = N[\ell_{\rho_{i \rightarrow j}(e)}(v)] \sqcup \left(\bigsqcup_{u \in N_{G_{\rho_{i \rightarrow j}(e)}}(v)} \text{new}N(u, c(u)) \right)$. Por definición de N_e y como $N_{G_e}(v) = N_{G_{\rho_{i \rightarrow j}(e)}}(v)$ para todo vértice v , tenemos $n = n'$ y por lo tanto $\text{check}_{v, c(v)}(n')$ es verdadero para cada $v \in V(G_e)$. Además, $w(\rho_{i \rightarrow j}(e), c, N) = w(e, c, N_e)$.

Concluimos probando que $\lambda(\rho_{i \rightarrow j}(e), C, N) \leq \alpha$. Sea C_e un vector en $\mathcal{N}^k$ que satisface (b), (c), y es tal que $\lambda(e, C_e, N_e) \neq \text{ERROR}$. Sea c un (C_e, N_e) -coloreo de G_e de peso $\lambda(e, C_e, N_e)$. Vamos a probar que c es también un (C, N) -coloreo de $G_{\rho_{i \rightarrow j}(e)}$. La condición (C1) es trivialmente satisfecha para cada etiqueta h en $\llbracket 1, k \rrbracket - \{i, j\}$. Para la etiqueta i , la condición (C1) también vale porque $C[i] = I$ y no hay vértices con etiqueta i en $V(G_{\rho_{i \rightarrow j}(e)})$. De manera similar a lo expuesto más arriba, podemos ver que $C[j] = \bigsqcup_{v \in \ell_{\rho_{i \rightarrow j}(e)}^{-1}(j)} \text{new}N(v, c(v))$. Sólo resta verificar la condición (C2), i.e., para cada vértice v , tenemos que $\text{check}_{v, c(v)}(n)$ es verdadero, donde $n = N[\ell_{\rho_{i \rightarrow j}(e)}(v)] \sqcup \left(\bigsqcup_{u \in N_{G_{\rho_{i \rightarrow j}(e)}}(v)} \text{new}N(u, c(u)) \right)$. Como antes, esto es consecuencia del hecho de que c es un (C_e, N_e) -coloreo de G_e y que, para cada vértice v , tenemos $N_e[\ell_e(v)] = N[\ell_{\rho_{i \rightarrow j}(e)}(v)]$ por definición y $N_{G_e}(v) = N_{G_{\rho_{i \rightarrow j}(e)}}(v)$. Por último, $w(\rho_{i \rightarrow j}(e), c, N) = w(e, c, N_e)$. $\square$

En el algoritmo, la entrada es la misma que en un problema 1-localmente verificable más un sistema de vecindades parciales universal y una k -expresión clique-width irredundante e_G del grafo de entrada G junto con su árbol T_{e_G} asociado, y su salida es el mínimo peso de un coloreo propio de G . Como explicamos anteriormente (y como se muestra en el Algoritmo 1), procedemos de abajo hacia arriba (bottom-up), es decir,

Algoritmo 1: Algoritmo principal

```

1 para cada subexpresión  $e$  de  $e_G$ , recorriendo el árbol de subexpresiones de
  abajo hacia arriba, hacer
2   para todos los vectores  $C, N \in \mathcal{N}^k$  hacer
3     Calcular  $\lambda(e, C, N)$  aplicando los Lemas 4.1.3, 4.1.4, 4.1.5 or 4.1.6, de
4     acuerdo al tipo de operación en la raíz de  $T_e$ .
5   Guardar el resultado en memoria.
6 Sea  $N_0$  el vector en  $\mathcal{N}^k$  tal que todos sus elementos son  $I$ .
7 Sea  $m \leftarrow \text{ERROR}$ .
8 para todo  $C \in \mathcal{N}^k$  tal que  $C[i] = I$  para cada  $i \in \overline{\ell(e_G)}$  hacer
9    $m \leftarrow \min(m, \lambda(e_G, C, N_0))$ 
10 devolver  $m$ 

```

empezamos con las hojas del árbol T_{e_G} , luego seguimos con sus padres y así sucesivamente, en cada caso calculando y almacenando en memoria el valor de $\lambda(e, C, N)$ para la subexpresión e correspondiente (aquella que corresponde al nodo de T_{e_G} que estamos analizando) y todas las posibles elecciones de C y N , usando las recurrencias dadas en los Lemas 4.1.3, 4.1.4, 4.1.5 y 4.1.6. Dado que estamos almacenando los resultados en memoria, la cantidad de veces que necesitamos calcular un valor $\lambda(\cdot, \cdot, \cdot)$ está dada por la cantidad de subexpresiones de e_G multiplicado por la cantidad de posibles vectores C y N , y como hay $O(|V(G)| + |E(G)|)$ subexpresiones de la expresión clique-width dada (ver Sección 1.6.2) y existen $|\mathcal{N}|^k$ vectores C posibles, respectivamente vectores N , este número es $O((|V(G)| + |E(G)|)\mathcal{N}^{2k})$. Por otro lado, determinar el mínimo entre todos los valores $\lambda(e_G, C, N_0)$, donde N_0 es el vector cuyos elementos son todos I y $C \in \mathcal{N}^k$ es cualquier vector tal que $C[i] = I$ para todo $i \in \overline{\ell(e_G)}$, se puede hacer en tiempo $O(|\mathcal{N}|^k)$.

Resta determinar la complejidad de calcular algún valor concreto de $\lambda(\cdot, \cdot, \cdot)$. Evidentemente, esto depende del tipo de operación considerada, así que distinguiremos los siguientes cuatro casos. Sean t_{check} , t_w , $t_{\boxplus}$, t_{newN} , $t_{\oplus}$, y $t_{\min}$ cotas superiores de los tiempos de ejecución de todas las funciones $check_{v,a}$, $w_{v,a}$, $\boxplus$, $newN$, $\oplus$, y $\min$, respectivamente. Sea $\mathcal{C} = \max\{|L_v| : v \in V(G)\}$. Vamos a suponer que otras operaciones son de tiempo constante.

- **Creación de un nuevo vértice:** Primero tenemos que recorrer (casi) todas las entradas de C , lo cual se puede hacer en tiempo $O(k)$. Luego, para cada $a \in L_v$, aplicamos una vez $newN$, $check_{v,a}$ y $w_{v,a}$, y almacenamos el mínimo peso. Así, obtenemos una complejidad $O(k + \mathcal{C}(t_{newN} + t_{check} + t_w + t_{\min}))$ para esta operación.
- **Unión disjunta:** Primero tenemos que determinar N_1 y N_2 , lo cual se puede hacer en tiempo $O(k)$, y luego tenemos que encontrar el peso mínimo iterando entre todas las posibles elecciones de C_1 y C_2 , lo cual se puede hacer en tiempo $O(|\mathcal{N}|^{2k})$, y además para cada elección estamos aplicando la operación $\boxplus$ una vez por cada etiqueta. Esto nos da una complejidad total $O(|\mathcal{N}|^{2k}(kt_{\boxplus} + t_{\oplus} + t_{\min}))$ en este caso.
- **Unión completa:** Simplemente tenemos que determinar la matriz N_e , lo cual se

puede hacer en tiempo $O(k + t_{\boxplus})$.

- **Renombre de etiqueta:** Primero debemos determinar la matriz N_e , lo cual se puede hacer en tiempo $O(k)$, y luego debemos encontrar el peso mínimo iterando sobre todas las posibles elecciones de C_e con todas las entradas fijas excepto dos de ellas. Luego, en este caso la complejidad resulta $O(k + |\mathcal{N}|^2(t_{\boxplus} + t_{\min}))$.

Finalmente, la complejidad de calcular cualquier $\lambda(e, C, N)$ está acotada por la suma de las complejidades de los cuatro casos, para lo cual obtenemos $O(\mathcal{C}(t_{newN} + t_{check} + t_w + t_{\min}) + |\mathcal{N}|^{2k}(kt_{\boxplus} + t_{\oplus} + t_{\min}))$. Así, la complejidad total de algoritmo es:

$$O((|V(G)| + |E(G)|)|\mathcal{N}|^{2k}(\mathcal{C}(t_{newN} + t_{check} + t_w + t_{\min}) + |\mathcal{N}|^{2k}(kt_{\boxplus} + t_{\oplus} + t_{\min}))).$$

Si el sistema de vecindades parciales está acotado por algún $\mathcal{P}$, podemos simplificar esta expresión obteniendo una complejidad en $O(k\mathcal{P}^{4k+1}\mathcal{C}(|V(G)| + |E(G)|))$.

Observación 4.1.7. Podemos modificar el algoritmo de modo de obtener también el coloreo como parte de la salida. Esto no afecta a la complejidad.

A continuación expondremos las principales consecuencias de este resultado. Como explicamos en la Sección 1.6.2, dado un grafo de clique-width k podemos encontrar una $f(k)$ -expresión clique-width irredundante en tiempo polinomial, por lo tanto no es realmente necesario contar con una expresión dada en la entrada (a menos que se quiera reducir la complejidad).

Teorema 4.1.8. *Sea Π un problema 1-localmente verificable con un sistema de vecindades parciales universal polinomial y con cantidad polinomial de colores. Luego, Π es XP parametrizado por clique-width. Más aún, si el sistema de vecindades parciales universal es constante entonces es FPT parametrizado por clique-width, y además es FPT lineal parametrizado por k si contamos con una k -expresión clique-width irredundante y $|L_v|$ está acotado por una constante para todo vértice v .*

Más precisamente, si el sistema de vecindades parciales está acotado por $\mathcal{P}$ entonces existe un algoritmo que resuelve el problema en grafos con una expresión clique-width irredundante dada en tiempo $O(w\mathcal{P}^{4w+1}\mathcal{C}(n + m))$ donde n y m son, respectivamente, la cantidad de vértices y de aristas del grafo de entrada, w es la clique-width de la expresión dada, y $\mathcal{C} = \max\{|L_v| : v \in V(G)\}$.

Los resultados que siguen se deducen a partir de las propiedades expuestas en la Sección 2.4 sobre las relaciones entre funciones contadoras de colores y sistemas de vecindades parciales.

Corolario 4.1.9. *Sea Π un problema 1-localmente verificable con cantidad constante de colores y funciones contadoras de colores. Luego, Π es XP parametrizado por clique-width.*

Corolario 4.1.10. *Sea Π un problema 1-localmente verificable con funciones d -estables o m -modulares, para algunas constantes $d, m \in \mathbb{N}$, y con cantidad de colores logarítmica en el tamaño del grafo de entrada (es decir, $O(\log |V(G)|)$, siendo G el grafo de entrada). Luego, Π es XP parametrizado por clique-width.*

Corolario 4.1.11. *Sea Π un problema 1-localmente verificable con cantidad constante de colores y funciones de verificación y peso que son d -estables o m -modulares, para algunas constantes $d, m \in \mathbb{N}$. Luego, Π es FPT parametrizado por clique-width. Más aún, si la entrada incluye una k -expresión clique-width irredundante, entonces es FPT lineal parametrizado por k .*

4.2. Propiedades globales

En esta sección presentaremos cómo extender el algoritmo de la sección anterior para así incluir propiedades globales de composición de condiciones locales. Procederemos de forma muy similar a como lo hicimos en la Sección 3.2, esto es, incorporando parámetros y definiendo recursivamente los $(C, N, p_1, \dots, p_m)$ -coloreos en G_e , para toda subexpresión e de la expresión e_G del grafo G de entrada. El caso base es aquel con $m = 0$ (cuyo algoritmo fue dado en la sección anterior). Luego, si $C, N \in \mathcal{N}^k$ y $p_1, \dots, p_m$ son parámetros tales que los $(C, N, p_1, \dots, p_m)$ -coloreos de G_e están definidos, entonces, para parámetros adicionales $p_{m+1}, \dots, p_{m+m'}$, definimos un $(C, N, p_1, \dots, p_m, p_{m+1}, \dots, p_{m+m'})$ -coloreo de G_e como un $(C, N, p_1, \dots, p_m)$ -coloreo de G_e tal que los parámetros $p_{m+1}, \dots, p_{m+m'}$ satisfacen alguna propiedad determinada. Nuevamente, definimos el peso de un $(C, N, p_1, \dots, p_m)$ coloreo c como $w(e, c, N)$ y denotaremos $\lambda(e, C, N, p_1, \dots, p_m)$ al peso mínimo entre todos los $(C, N, p_1, \dots, p_m)$ -coloreos de G_e .

Al igual que antes, en lo que sigue escribiremos $\hat{p}$ en lugar de $p_1, \dots, p_m$ para simplificar la notación. Recordemos que $\hat{p}$ es vacío cuando $m = 0$.

4.2.1. Composición de condiciones locales

Consideremos un problema 1-localmente verificable Π con un sistema de vecindades parciales universal, un conjunto de propiedades globales Γ y un grafo de entrada G con una k -expresión clique-width e_G de G .

Sean $(M, \otimes)$ un monoide conmutativo, X un subconjunto de M y f una función que recibe un vértice v y una asignación de colores válida en $N_G[v]$ y devuelve un elemento de M . Sea o el elemento neutro de M y sea $\in_X: M \rightarrow \text{BOOL}$ la función tal que $\in_X(x) = (x \in X)$ para todo $x \in M$. Supongamos que, para cada $v \in V(G)$ y $i \in L_v$, existe una función $f_{v,i}: \mathcal{N} \rightarrow M$ tal que $f_{v,c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) = f(v, c)$ para cada vértice $v \in V(G)$ y cada coloreo c válido en $N_G[v]$.

Veremos cómo obtener el mínimo peso de un coloreo propio c que satisfaga Γ y además sea tal que $\bigotimes_{v \in V(G)} f(v, c) \in X$, es decir, $\bigotimes_{v \in V(G)} f_{v,c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) \in X$. Para ello vamos a considerar el siguiente parámetro adicional: un elemento $m \in M$, que intuitivamente va a representar cuánto queremos acumular en el subgrafo G_e . Definimos un $(C, N, \hat{p}, m)$ -coloreo c de G_e como un $(C, N, \hat{p})$ -coloreo de G_e tal que $m = \bigotimes_{v \in V(G_e)} f_{v,c(v)} \left(N[\ell_e(v)] \bigoplus \bigoplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right)$. Como mencionamos anteriormente, vamos a notar $\lambda(e, C, N, \hat{p}, m)$ al peso mínimo entre todos los $(C, N, \hat{p}, m)$ -coloreos de G_e .

Lema 4.2.1. *Sea $N_0 \in \mathcal{N}^k$ el vector cuyos elementos son todos iguales a I . Supongamos que el mínimo peso de un coloreo propio de G que satisface las propiedades globales Γ es igual a*

$$\min\{\lambda(e_G, C, N_0, \hat{p}) : P(C, \hat{p}) = \text{TRUE}\}$$

para alguna propiedad P que implica aquella del Lema 4.1.2. Más aún, supongamos que

- (1) para cada coloreo propio c de G que satisface Γ existen $C, \hat{p}$ tales que $P(C, \hat{p}) = \text{TRUE}$ y tales que c es un $(C, N_0, \hat{p})$ -coloreo de G ;

- (2) para cada $C, \hat{p}$ tales que $P(C, \hat{p}) = \text{TRUE}$ y tales que $\lambda(e_G, C, N_0, \hat{p}) \neq \text{ERROR}$, cada $(C, N_0, \hat{p})$ -coloreo de G es un coloreo propio de G que satisface Γ .

Luego, el mínimo peso de un coloreo propio c de G que satisface Γ y tal que $\bigotimes_{v \in V(G)} f_{v, c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) \in X$ es igual a

$$\min\{\lambda(e_G, C, N_0, \hat{p}, m) : P(C, \hat{p}) = \text{TRUE} \text{ y } m \in X\}.$$

Más aún,

- (a) para cada coloreo propio c de G que satisface Γ y es tal que $\bigotimes_{v \in V(G)} f_{v, c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) \in X$, existen $C, \hat{p}$ y $m \in M$ tales que $P(C, \hat{p}) = \text{TRUE}$ y c es un $(C, N_0, \hat{p}, m)$ -coloreo de G ,
- (b) para cada $C, \hat{p}$ y $m \in M$ tales que $P(C, \hat{p}) = \text{TRUE}$ y $\lambda(e_G, C, N_0, \hat{p}, m) \neq \text{ERROR}$, cada $(C, N_0, \hat{p}, m)$ -coloreo c de G es un coloreo propio de G que satisface Γ y además $\bigotimes_{v \in V(G)} f_{v, c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) \in X$.

Demostración. Supongamos que tenemos un coloreo propio c de G que satisface Γ y tal que $n \in X$ donde $n = \bigotimes_{v \in V(G)} f_{v, c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right)$. Vamos a probar que existen $C, \hat{p}$ y $m \in M$ tales que $P(C, \hat{p}) = \text{TRUE}$, c es un $(C, N_0, \hat{p}, m)$ -coloreo de G y $w(c) \geq \lambda(e_G, C, N_0, \hat{p}, m)$. Por la hipótesis (1), sabemos que existen $C, \hat{p}$ tales que $P(C, \hat{p}) = \text{TRUE}$ y c es un $(C, N_0, \hat{p})$ -coloreo de G . Tomemos $m = n$. Es fácil ver que $n = \bigotimes_{v \in V(G)} f_{v, c(v)} \left(N_0[\ell(v)] \boxplus \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) \right)$, porque $N_0[\ell(v)]$ es el neutro de $\boxplus$ para todo $v \in V(G)$. Por lo tanto se cumple la condición extra para ser un $(C, N_0, \hat{p}, m)$ -coloreo de G . Luego, c es un $(C, N_0, \hat{p}, m)$ -coloreo de G y $w(c) = w(e_G, c, N_0) \geq \lambda(C, N_0, \hat{p}, m)$.

Supongamos ahora que tenemos $C, \hat{p}$ y $m \in M$ tales que $P(C, \hat{p}) = \text{TRUE}$ y tales que $\lambda(C, N_0, \hat{p}, m) \neq \text{ERROR}$. En particular, esto implica que $m \in X$. Sea c un $(C, N_0, \hat{p}, m)$ -coloreo de G (sabemos que existe al menos uno). Por definición, c es un $(C, N_0, \hat{p})$ -coloreo de G , y por la hipótesis (2), c es también un coloreo propio de G que satisface Γ . Por otro lado, nuevamente por definición de $(C, N_0, \hat{p}, m)$ -coloreo, sabemos que se verifica que $m = \bigotimes_{v \in V(G)} f_{v, c(v)} \left(N_0[\ell(v)] \boxplus \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right) \right) = \bigotimes_{v \in V(G)} f_{v, c(v)} \left(\bigoplus_{u \in N_G(v)} \text{new}N(u, c(u)) \right)$. Como $m \in X$, estamos. Si consideramos en particular un c de peso mínimo, entonces $\lambda(e_G, C, N_0, \hat{p}, m) = w(e_G, c, N_0) = w(c)$.

Notemos que (a) y (b) quedan implícitamente demostrados por lo anterior. $\square$

Para los siguientes lemas, vamos a suponer que contamos con vectores C y N en $\mathcal{N}^k$, una lista de parámetros (posiblemente vacía) $\hat{p}$ para los cuales ya tenemos definidos los $(C, N, \hat{p})$ -coloreos, y $m \in M$.

Lema 4.2.2 (Creación de un nuevo vértice: $i(v)$). *Supongamos que*

$$\lambda(i(v), C, N, \hat{p}) = \min\{w_{v,a}(N[i]) : P(a, C, N, \hat{p}) = \text{TRUE}\}.$$

para alguna propiedad P que implica aquella del Lema 4.1.3. Más aún, supongamos que c es un $(C, N, \hat{p})$ -coloreo de $G_{i(v)}$ si y sólo si c es un coloreo de $G_{i(v)}$ tal que $P(c(v), C, N, \hat{p}) = \text{TRUE}$. Entonces,

$$\lambda(i(v), C, N, \hat{p}, m) = \min\{w_{v,a}(N[i]) : P(a, C, N, \hat{p}) = \text{TRUE} \text{ y } m = f_{v,a}(N[i])\}.$$

Más aún, c es un $(C, N, \hat{p}, m)$ -coloreo de $G_{i(v)}$ si y sólo si c es un coloreo de $G_{i(v)}$ tal que $P(c(v), C, N, \hat{p}) = \text{TRUE}$ y $m = f_{v,c(v)}(N[i])$.

Demostración. Sabemos que $G_{i(v)}$ es un grafo que contiene un único vértice v , el cual tiene etiqueta i . Por definición, c es un $(C, N, \hat{p}, m)$ -coloreo si y sólo si c es un $(C, N, \hat{p})$ -coloreo de $G_{i(v)}$ tal que $m = \bigotimes_{w \in V(G_{i(v)})} f_{w,c(w)} \left(N[\ell_{i(v)}(w)] \boxplus \bigoplus_{u \in N_{G_{i(v)}}(w)} \text{new}N(u, c(u)) \right) = f_{v,c(v)}(N[i])$. Por lo tanto, c es un $(C, N, \hat{p}, m)$ -coloreo si y sólo si c es un $(C, N, \hat{p})$ -coloreo de $G_{i(v)}$ tal que $m = f_{v,c(v)}(N[i])$, y el enunciado sigue trivialmente. $\square$

Lema 4.2.3 (Unión disjunta: $e_1 \oplus e_2$). *Supongamos que*

$$\lambda(e_1 \oplus e_2, C, N, \hat{p}) = \min\{\lambda(e_1, C_1, N_1, \hat{p}_1) \oplus \lambda(e_2, C_2, N_2, \hat{p}_2) : \\ P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}\}$$

para alguna propiedad P que implica aquella del Lema 4.1.4. Más aún, supongamos que

- (1) para cada $(C, N, \hat{p})$ -coloreo c de $G_{e_1 \oplus e_2}$ existen $C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ tales que $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$ y tales que $c|_{V(G_{e_1})}$ es un $(C_1, N_1, \hat{p}_1)$ -coloreo de G_{e_1} y $c|_{V(G_{e_2})}$ es un $(C_2, N_2, \hat{p}_2)$ -coloreo de G_{e_2} ;
- (2) para todos $C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ tales que $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$, si c_1 es un $(C_1, N_1, \hat{p}_1)$ -coloreo de G_{e_1} y c_2 es un $(C_2, N_2, \hat{p}_2)$ -coloreo de G_{e_2} , entonces $c = c_1 \cup c_2$ es un $(C, N, \hat{p})$ -coloreo de $G_{e_1 \oplus e_2}$.

Entonces,

$$\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) = \min\{\lambda(e_1, C_1, N_1, \hat{p}_1, m_1) \oplus \lambda(e_2, C_2, N_2, \hat{p}_2, m_2) : \\ P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}, \\ m_1, m_2 \in M \text{ y } m_1 \otimes m_2 = m\}.$$

Más aún,

- (a) para cada $(C, N, \hat{p}, m)$ -coloreo c de $G_{e_1 \oplus e_2}$ existen $m_1, m_2, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ tales que $m_1, m_2 \in M$, $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$, $m_1 \otimes m_2 = m$, y $c|_{V(G_{e_1})}$ es un $(C_1, N_1, \hat{p}_1, m_1)$ -coloreo de G_{e_1} y $c|_{V(G_{e_2})}$ es un $(C_2, N_2, \hat{p}_2, m_2)$ -coloreo de G_{e_2} ;
- (b) para todos $m_1, m_2, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ tales que $m_1, m_2 \in M$, $m_1 \otimes m_2 = m$ y $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$, si c_1 es un $(C_1, N_1, \hat{p}_1, m_1)$ -coloreo de G_{e_1} y c_2 es un $(C_2, N_2, \hat{p}_2, m_2)$ -coloreo de G_{e_2} entonces $c = c_1 \cup c_2$ es un $(C, N, \hat{p}, m)$ -coloreo de $G_{e_1 \oplus e_2}$.

Demostración. Sea $\alpha = \min\{\lambda(e_1, C_1, N_1, \hat{p}_1, m_1) \oplus \lambda(e_2, C_2, N_2, \hat{p}_2, m_2) : P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}, m_1, m_2 \in M \text{ y } m_1 \otimes m_2 = m\}$. Primero vamos a demostrar que $\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) \geq \alpha$. Notemos que si $\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) = \text{ERROR}$, entonces terminamos. Por lo tanto supongamos que $\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) \neq \text{ERROR}$. Sea c un $(C, N, \hat{p}, m)$ -coloreo de $G_{e_1 \oplus e_2}$. En otras palabras, c es un $(C, N, \hat{p})$ -coloreo de $G_{e_1 \oplus e_2}$ tal que $m = \bigotimes_{v \in V(G_{e_1 \oplus e_2})} f_{v,c(v)} \left(N[\ell_{e_1 \oplus e_2}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{e_1 \oplus e_2}}(v)} \text{new}N(u, c(u)) \right) \right)$. Sean

$c_1 = c|_{V(G_{e_1})}$ y $c_2 = c|_{V(G_{e_2})}$. Vamos a mostrar que existen $C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ y $m_1, m_2 \in M$ tales que $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$, $m_1 \otimes m_2 = m$, y c_1 es un $(C_1, N_1, \hat{p}_1, m_1)$ -coloreo de G_{e_1} , y c_2 es un $(C_2, N_2, \hat{p}_2, m_2)$ -coloreo de G_{e_2} . Por la hipótesis (1), ya sabemos que existen $C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ tales que $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$, y c_1 es un $(C_1, N_1, \hat{p}_1)$ -coloreo de G_{e_1} y c_2 es un $(C_2, N_2, \hat{p}_2)$ -coloreo de G_{e_2} . Para cada $i \in \{1, 2\}$, sea $m_i = \bigotimes_{v \in V(G_{e_i})} f_{v, c(v)} \left(N_i[\ell_{e_i}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{e_i}}(v)} \text{new}N(u, c(u)) \right) \right)$. Claramente, $m_1 \otimes m_2 = m$, ya que $V(G_{e_1 \oplus e_2}) = V(G_{e_1}) \cup V(G_{e_2})$, $V(G_{e_1})$ y $V(G_{e_2})$ son disjuntos, y $N_1[i] = N[i]$ para toda etiqueta $i \notin \ell(e_1)$, y análogamente para N_2 (ver Lema 4.1.4). Luego, c_1 es un $(C_1, N_1, \hat{p}_1, m_1)$ -coloreo de G_{e_1} y c_2 es un $(C_2, N_2, \hat{p}_2, m_2)$ -coloreo de G_{e_2} . Si consideramos en particular un $(C, N, \hat{p}, m)$ -coloreo c de $G_{e_1 \oplus e_2}$ de peso mínimo, obtenemos $\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) = w(e_1 \oplus e_2, c, N) = w(e_1, c_1, N_1) \oplus w(e_2, c_2, N_2) \geq \lambda(e_1, C_1, N_1, \hat{p}_1, m_1) \oplus \lambda(e_2, C_2, N_2, \hat{p}_2, m_2) \geq \alpha$.

Ahora vamos a probar que $\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) \leq \lambda(e_1, C_1, N_1, \hat{p}_1, m_1) \oplus \lambda(e_2, C_2, N_2, \hat{p}_2, m_2)$ siempre que $m_1, m_2 \in M$, $m_1 \otimes m_2 = m$ y $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$. Esto implica directamente que $\lambda(e_1 \oplus e_2, C, N, \hat{p}, m) \leq \alpha$. Sean $m_1, m_2 \in M$ y $C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2$ tales que $m_1 \otimes m_2 = m$ y $P(C, N, \hat{p}, C_1, N_1, \hat{p}_1, C_2, N_2, \hat{p}_2) = \text{TRUE}$. Si $\lambda(e_1, C_1, N_1, \hat{p}_1, m_1) = \text{ERROR}$ o $\lambda(e_2, C_2, N_2, \hat{p}_2, m_2) = \text{ERROR}$, entonces terminamos. Así que supongamos que $\lambda(e_1, C_1, N_1, \hat{p}_1, m_1) \neq \text{ERROR}$ y $\lambda(e_2, C_2, N_2, \hat{p}_2, m_2) \neq \text{ERROR}$. Sean c_1 un $(C_1, N_1, \hat{p}_1, m_1)$ -coloreo de G_{e_1} y c_2 un $(C_2, N_2, \hat{p}_2, m_2)$ -coloreo de G_{e_2} . Para cada $i \in \{1, 2\}$, sabemos que $m_i = \bigotimes_{v \in V(G_{e_i})} f_{v, c_i(v)} \left(N_i[\ell_{e_i}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{e_i}}(v)} \text{new}N(u, c_i(u)) \right) \right)$. Consideremos $c = c_1 \cup c_2$ el cual, por la hipótesis (2), es un $(C, N, \hat{p})$ -coloreo de $G_{e_1 \oplus e_2}$. Tenemos $w(e_1 \oplus e_2, c, N) = w(e_1, c_1, N_1) \oplus w(e_2, c_2, N_2)$. Para mostrar que c es un $(C, N, \hat{p}, m)$ -coloreo de $G_{e_1 \oplus e_2}$, resta probar que $m = n$, donde $n = \bigotimes_{v \in V(G_{e_1 \oplus e_2})} f_{v, c(v)} \left(N[\ell_{e_1 \oplus e_2}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{e_1 \oplus e_2}}(v)} \text{new}N(u, c(u)) \right) \right)$. Naturalmente, $n = m_1 \otimes m_2$, ya que $V(G_{e_1 \oplus e_2}) = V(G_{e_1}) \cup V(G_{e_2})$, $V(G_{e_1})$ y $V(G_{e_2})$ son disjuntos, y $N_1[i] = N[i]$ para toda etiqueta $i \notin \ell(e_1)$, y análogamente para N_2 (ver Lema 4.1.4). Entonces $m = n$. Así, c es un $(C, N, \hat{p}, m)$ -coloreo de $G_{e_1 \oplus e_2}$. Si consideramos en particular un $(C_1, N_1, \hat{p}_1, m_1)$ -coloreo c_1 de G_{e_1} de peso mínimo así como también un $(C_2, N_2, \hat{p}_2, m_2)$ -coloreo c_2 de G_{e_2} de peso mínimo, obtenemos $\lambda(e_1, C_1, N_1, \hat{p}_1, m_1) \oplus \lambda(e_2, C_2, N_2, \hat{p}_2, m_2) = w(e_1, c_1, N_1) \oplus w(e_2, c_2, N_2) = w(e_1 \oplus e_2, c, N) \geq \lambda(e_1 \oplus e_2, C, N, \hat{p}, m)$.

Notemos que (a) y (b) quedan implícitamente demostrados por lo anterior. $\square$

Lema 4.2.4 (Unión completa: $\eta_{i,j}(e)$). *Sea N' definido como en el Lema 4.1.5. Supongamos que existe $\hat{p}'$ tal que c es un $(C, N, \hat{p})$ -coloreo de $G_{\eta_{i,j}(e)}$ si y sólo si c es un $(C, N', \hat{p}')$ -coloreo de G_e .*

Luego, c es un $(C, N, \hat{p}, m)$ -coloreo de $G_{\eta_{i,j}(e)}$ si y sólo si c es un $(C, N', \hat{p}', m)$ -coloreo de G_e . En particular,

$$\lambda(\eta_{i,j}(e), C, N, \hat{p}, m) = \lambda(e, C, N', \hat{p}', m).$$

Demostración. Sea c un $(C, N, \hat{p}, m)$ -coloreo de $G_{\eta_{i,j}(e)}$. Directamente a partir de la definición, podemos decir que c es un $(C, N, \hat{p})$ -coloreo de $G_{\eta_{i,j}(e)}$ tal que $m =$

$$\bigotimes_{v \in V(G_{\eta_{i,j}(e)})} f_{v, c(v)} \left(N[\ell_{\eta_{i,j}(e)}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{\eta_{i,j}(e)}}(v)} \text{new}N(u, c(u)) \right) \right).$$

Por hipótesis,

sabemos que existe $\widehat{p}'$ tal que c es también un $(C, N', \widehat{p}')$ -coloreo de G_e . Resta probar que $m = n$ donde $n = \bigotimes_{v \in V(G_e)} f_{v,c(v)} \left(N'[\ell_e(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \right)$. Por definición de N' , tenemos que $N[\ell_{\eta_{i,j}(e)}(v)] \boxplus \left(\boxplus_{u \in N_{G_{\eta_{i,j}(e)}}(v)} \text{new}N(u, c(u)) \right) = N'[\ell_e(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right)$ para todo vértice $v \in V(G_{\eta_{i,j}(e)}) = V(G_e)$. Entonces $m = n$ y c resulta ser también un $(C, N', \widehat{p}', m)$ -coloreo de G_e . De forma similar podemos probar la vuelta. $\square$

Lema 4.2.5 (Renombre de etiqueta: $\rho_{i \rightarrow j}$). *Supongamos que*

$$\lambda(\rho_{i \rightarrow j}(e), C, N, \widehat{p}) = \min\{\lambda(e, C_e, N_e, \widehat{p}_e) : P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}\}$$

para alguna propiedad P que implica aquella del Lema 4.1.6. Más aún, supongamos que

- (1) para cada $(C, N, \widehat{p})$ -coloreo c de $G_{\rho_{i \rightarrow j}(e)}$, existen parámetros $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$ y c es un $(C_e, N_e, \widehat{p}_e)$ -coloreo de G_e ;
- (2) para todos los parámetros $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$, si c es un $(C_e, N_e, \widehat{p}_e)$ -coloreo de G_e , entonces c es también un $(C, N, \widehat{p})$ -coloreo de $G_{\rho_{i \rightarrow j}(e)}$.

Luego,

$$\lambda(\rho_{i \rightarrow j}(e), C, N, \widehat{p}, m) = \min\{\lambda(e, C_e, N_e, \widehat{p}_e, m) : P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}\}.$$

Más aún,

- (a) para cada $(C, N, \widehat{p}, m)$ -coloreo c de $G_{\rho_{i \rightarrow j}(e)}$, existen parámetros $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$ y c es un $(C_e, N_e, \widehat{p}_e, m)$ -coloreo de G_e ;
- (b) para todos los parámetros $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$, si c es un $(C_e, N_e, \widehat{p}_e, m)$ -coloreo de G_e , entonces c es también un $(C, N, \widehat{p}, m)$ -coloreo de $G_{\rho_{i \rightarrow j}(e)}$.

Demostración. Sea $\alpha = \min\{\lambda(e, C_e, N_e, \widehat{p}_e, m) : P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}\}$. Vamos a probar primero que $\lambda(\rho_{i \rightarrow j}(e), C, N, \widehat{p}, m) \geq \alpha$. Sea c un $(C, N, \widehat{p}, m)$ -coloreo de $G_{\rho_{i \rightarrow j}(e)}$. Vamos a mostrar que existen parámetros $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$ y c es un $(C_e, N_e, \widehat{p}_e, m)$ -coloreo de G_e . Por definición, c es un $(C, N, \widehat{p})$ -coloreo de $G_{\rho_{i \rightarrow j}(e)}$ tal que $m = \bigotimes_{v \in V(G_{\rho_{i \rightarrow j}(e)})} f_{v,c(v)} \left(N[\ell_{\rho_{i \rightarrow j}(e)}(v)] \boxplus \left(\boxplus_{u \in N_{G_{\rho_{i \rightarrow j}(e)}}(v)} \text{new}N(u, c(u)) \right) \right)$. Por lo tanto, por la hipótesis (1), existen parámetros $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$ y c es un $(C_e, N_e, \widehat{p}_e)$ -coloreo de G_e . Como vimos en la demostración del Lema 4.1.6, la matriz N_e es tal que $N_e[\ell_e(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) = N[\ell_{\rho_{i \rightarrow j}(e)}(v)] \boxplus \left(\boxplus_{u \in N_{G_{\rho_{i \rightarrow j}(e)}}(v)} \text{new}N(u, c(u)) \right)$ para todo vértice $v \in V(G_e) = V(G_{\rho_{i \rightarrow j}(e)})$. Entonces, si definimos $n_e = \bigotimes_{v \in V(G_e)} f_{v,c(v)} \left(N_e[\ell_e(v)] \boxplus \left(\boxplus_{u \in N_{G_e}(v)} \text{new}N(u, c(u)) \right) \right)$ tenemos que $n_e = m$. Por lo tanto, c es un $(C_e, N_e, \widehat{p}_e, m)$ -coloreo de G_e . Si consideramos en particular un

$(C, N, \widehat{p}, m)$ -coloreo c de $G_{\rho_{i \rightarrow j}(e)}$ de peso mínimo, entonces $\lambda(\rho_{i \rightarrow j}(e), C, N, \widehat{p}, m) = w(\rho_{i \rightarrow j}(e), c, N) = w(e, c, N_e) \geq \lambda(e, C_e, N_e, \widehat{p}_e, m) \geq \alpha$.

Veamos ahora que $\lambda(\rho_{i \rightarrow j}(e), C, N, \widehat{p}, m) \leq \alpha$. Sean $C_e, N_e, \widehat{p}_e$ tales que $P(C, N, \widehat{p}, C_e, N_e, \widehat{p}_e) = \text{TRUE}$. Si $\lambda(e, C_e, N_e, \widehat{p}_e, m) = \text{ERROR}$ entonces terminamos. Por lo tanto supongamos que $\lambda(e, C_e, N_e, \widehat{p}_e, m) \neq \text{ERROR}$ y sea c un $(C_e, N_e, \widehat{p}_e, m)$ -coloreo de G_e . Por definición, c es un $(C_e, N_e, \widehat{p}_e)$ -coloreo de G_e tal que $m = \bigotimes_{v \in V(G_e)} f_{v, c(v)} \left(N_e[\ell_e(v)] \boxplus \left(\bigoplus_{u \in N_{G_e}(v)} \text{new} N(u, c(u)) \right) \right)$. De acuerdo a la hipótesis (2), c es también un $(C, N, \widehat{p})$ -coloreo de $G_{\rho_{i \rightarrow j}(e)}$. Nuevamente según lo visto en la demostración del Lema 4.1.6, sabemos que el vector N_e es tal que $N_e[\ell_e(v)] \boxplus \left(\bigoplus_{u \in N_{G_e}(v)} \text{new} N(u, c(u)) \right) = N[\ell_{\rho_{i \rightarrow j}(e)}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{\rho_{i \rightarrow j}(e)}}(v)} \text{new} N(u, c(u)) \right)$ para todo vértice $v \in V(G_e) = V(G_{\rho_{i \rightarrow j}(e)})$. Si definimos $n = \bigotimes_{v \in V(G_{\rho_{i \rightarrow j}(e)})} f_{v, c(v)} \left(N[\ell_{\rho_{i \rightarrow j}(e)}(v)] \boxplus \left(\bigoplus_{u \in N_{G_{\rho_{i \rightarrow j}(e)}}(v)} \text{new} N(u, c(u)) \right) \right)$ entonces $n = m$. Sigue inmediatamente que c es un $(C, N, \widehat{p}, m)$ -coloreo de $G_{\rho_{i \rightarrow j}(e)}$. Si consideramos en particular un $(C_e, N_e, \widehat{p}_e, m)$ -coloreo c de G_e con el cual obtenemos el peso α , entonces $\alpha = w(e, c, N_e) = w(\rho_{i \rightarrow j}(e), c, N) \geq \lambda(\rho_{i \rightarrow j}(e), C, N, \widehat{p}, m)$.

Notemos que (a) y (b) quedan implícitamente demostrados por lo anterior. $\square$

Complejidad del algoritmo modificado

Supongamos que estamos aplicando esta propiedad global $\mathcal{R}$ veces, con monoides $(M_1, \otimes_1), \dots, (M_{\mathcal{R}}, \otimes_{\mathcal{R}})$, subconjuntos $X_1, \dots, X_{\mathcal{R}}$ y funciones $f^1, \dots, f^{\mathcal{R}}$. Sea $\mathcal{M} = \max_{i=1}^{\mathcal{R}} |M_i|$ y sea t_f una cota superior del tiempo para calcular cualquier función $f_{v,a}^i$. Vamos a suponer que podemos calcular $x \otimes y$ y $x \in X_i$ en tiempo constante para todos $i \in [1, \mathcal{R}]$, $x, y \in M_i$ y $X_i \subseteq M_i$. Luego, en cada caso la complejidad debe modificarse de la siguiente manera:

- **Creación de un nuevo vértice:** Sumamos $\mathcal{R} \cdot t_f$.
- **Unión disjunta:** Multiplicamos por $\mathcal{M}^{2\mathcal{R}}$.
- **Unión completa:** Sin cambios.
- **Renombre de etiqueta:** Sin cambios.

Cuando iteramos sobre todos los posibles $\lambda(e, C, N, \widehat{p}, m_1, \dots, m_{\mathcal{R}})$, debemos multiplicar la complejidad por un factor $\mathcal{M}^{\mathcal{R}}$, porque para cada m_i hay a lo sumo $\mathcal{M}$ valores posibles.

Por lo tanto, si tanto $\mathcal{M}^{\mathcal{R}}$ como t_f están acotados por un polinomio en $|V(G)|$, no se alteran los casos en los que el algoritmo que vimos en la Sección 4.1 es XP o FPT (manteniéndose FPT lineal sólo si $\mathcal{M}^{\mathcal{R}}$ y t_f son constantes).

Notemos que si el sistema de vecindades parciales universal propuesto para *check* y *w* “no funciona” para f (es decir, si no existen las funciones $f_{v,i}$ que cumplan lo pedido) pero sí existe otro sistema que “funciona” para f , entonces podemos considerar un nuevo sistema de vecindades parciales universal que contenga a ambos (esencialmente, cada uno de sus componentes se obtiene del producto cartesiano de los componentes de ambos sistemas).

En particular, si usamos esta propiedad global para fijar el tamaño de algún conjunto de clases de colores, entonces $\mathcal{M} \leq |V(G)|$. Además, como la función f en este caso ignora lo acumulado por los vecinos del vértice, cualquier sistema de vecindad parcial funciona para f y t_f es constante. Por lo tanto, si un problema Π admite un sistema de vecindades parciales universal polinomial (resp. constante), el problema Π con esta propiedad global es XP (resp. FPT) parametrizado por clique-width. Más aún, esto se mantiene si aplicamos esta propiedad una cantidad constante de veces (en particular, cuando la cantidad total de colores es constante podemos poner restricciones sobre cualquier combinación de conjuntos de clases de colores).

Notemos también que si usamos esta propiedad para fijar la cantidad de aristas entre vértices de algunas clases de colores, tenemos $\mathcal{M} \leq |E(G)|$.

4.3. El caso $r > 1$

Si en un problema r -localmente verificable las funciones de verificación y de peso no hacen ninguna distinción entre las cantidades de vecinos a las diferentes distancias entre 1 y r , entonces podemos reducir este problema a un problema 1-localmente verificable en la r -ésima potencia del grafo de entrada. Por la Proposición 3.1 en [40], para cada grafo G de clique-width a lo sumo k , la clique-width de G^r es a lo sumo $4(r+1)^k$. Entonces, para una distancia r constante, los resultados XP/FPT de la Sección 4.1 siguen valiendo.

Como veremos en el Capítulo 5, los problemas r -localmente verificables con cantidad constante de colores y funciones d -estables a t distancias $r_1, \dots, r_t$, con t constante, son XP parametrizados por mim-width, y por lo tanto también por clique-width. Queda abierta la complejidad parametrizada por clique-width de los problemas r -localmente verificables con funciones contadoras de colores (no-estables) a t distancias $r_1, \dots, r_t$ con $t > 1$, o con $t = 1$ pero r no constante. Es probable que para resolver estos casos sea necesario cambiar notablemente el algoritmo, trabajando sobre alguna expresión de G^{r_t} que permita reconocer la distancia en G entre los vértices que conectan mediante una operación $\eta_{i,j}$.

4.4. Límites

Tal y como hicimos en el Capítulo 3, en esta sección veremos que los resultados obtenidos son los mejores que podemos esperar, en el sentido de que si eliminamos alguna de las restricciones consideradas entonces nos encontramos con problemas para-NP-difíciles parametrizados por clique-width (en el caso de las restricciones para obtener algoritmos XP) y W[1]-difíciles (en el caso de las restricciones para obtener algoritmos FPT).

Si incluyéramos etiquetas en las aristas entonces podríamos reducir MÍNIMO CONJUNTO DOMINANTE en un grafo cualquiera G al siguiente problema 1-localmente verificable en el grafo completo G' con $V(G') = V(G)$:

- $\ell_{uv} = 1$ si $uv \in E(G)$ y $\ell_{uv} = 0$ si no,
- COLORS = $\{0, 1\}$,

- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w(v, c) = c(v)$, y
- $check(v, c) = \left(c(v) + \sum_{u \in N_{G'}(v)} (c(u) \cdot \ell_{vu}) \geq 1 \right)$.

Esto quiere decir que si pudiéramos resolver en tiempo polinomial cualquier problema 1-localmente verificable con etiquetas en las aristas (e incluso si nos restringimos a aquellos que tienen sólo 2 tipos de etiquetas y también 2 colores) en grafos completos entonces probaríamos que $P=NP$. Lo mismo ocurre si consideramos problemas sin etiquetas en las aristas pero con una función de verificación o de peso que, si bien admite un sistema de vecindades parciales constante, no admite uno universal polinomial. En efecto, tomemos la reducción anterior pero con el siguiente problema 1-localmente verificable en G' :

- $\text{COLORS} = \{0, 1\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w(v, c) = c(v)$, y
- $check(v, c) = \left(c(v) + \sum_{u \in N_G(v)} c(u) \geq 1 \right)$.

Aquí la función *check* usa información sobre qué vecinos en G' de un vértice v son además sus vecinos en G , por lo tanto no admite un sistema de vecindades parciales universal polinomial. Análogamente, si consideramos

- $\text{COLORS} = \{0, 1\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w(v, c) = c(v)$ si $c(v) + \sum_{u \in N_G(v)} c(u) \geq 1$, y $w(v, c) = +\infty$ si no, y
- $check(v, c) = \text{TRUE}$,

entonces también estamos modelando el mismo problema, pero con una función de peso que no admite un sistema de vecindades parciales universal polinomial. Notemos que este truco se puede aplicar siempre, por eso la función de peso tiene las mismas restricciones que la de verificación.

Ya vimos en la Sección 3.4 que el problema de ETIQUETADO $L(2, 1)$ se puede modelar con funciones 1-estables pero cantidad de colores polinomial en el grado máximo del grafo. Por lo tanto, no podemos eliminar la restricción sobre la cantidad de colores cuando tenemos funciones contadoras de colores. Esto ocurre incluso si consideramos funciones 2-modulares. Para probar esto, reduciremos a un problema 1-localmente verificable el problema de encontrar un mínimo conjunto dominante impar (el cual es un problema NP-difícil [161]). Para un grafo de entrada G , con $V(G) = \{v_1, \dots, v_n\}$, definimos el grafo completo G' con $V(G') = V(G)$. Para cada $i \in \llbracket 1, n \rrbracket$, sea $N_i = \{j \in \llbracket 1, n \rrbracket : v_j \in N_G(v_i)\}$. Consideremos el siguiente problema 1-localmente verificable con funciones 2-modulares a resolver en G' :

- $\text{COLORS} = \llbracket 0, n \rrbracket$,

- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w'(v_i, a, \mu) = \begin{cases} 1 & \text{si } a = i \\ 0 & \text{si no,} \end{cases}$
- $check'(v_i, a, \mu) = ((a = i \wedge \sum_{j \in N_i} \mu_j \equiv 0 \pmod{2}) \vee (a = 0 \wedge \sum_{j \in N_i} \mu_j \equiv 1 \pmod{2}))$.

A un coloreo propio c en G' le podemos asociar el conjunto $S = \{v_i \in V(G) : c(v_i) = i\}$. Veamos que S es en efecto un conjunto dominante impar en G . En primer lugar, para cada vértice $v_i \in S$ tenemos $c(v_i) = i$ y entonces, por definición de *check*, tenemos que la cantidad de vértices $v_j \in N_G(v_i)$ tales que $c(v_j) = j$ es par, lo que significa que $|N_G[v_i] \cap S| \equiv 1 \pmod{2}$. Por último, para cada vértice v_i que no está en S tenemos $c(v_i) = 0$ y entonces, por definición de *check*, tenemos que la cantidad de vértices $v_j \in N_G(v_i)$ tales que $c(v_j) = j$ es impar, lo que significa que $|N_G[v_i] \cap S| \equiv 1 \pmod{2}$. Como G' es un grafo completo, tiene clique-width igual a 2. En conclusión, no podemos aspirar a resolver eficientemente cualquier problema 1-localmente verificable con funciones 1-estables o 2-modulares y cantidad polinomial de colores en grafos completos.

Analicemos ahora las restricciones con las cuales obtenemos algoritmos FPT. Para ello vamos a analizar el problema MAX-CUT. Dado un grafo G , un *corte* es una partición de los vértices en dos conjuntos disjuntos S y T , y su tamaño se define como la cantidad de aristas en el grafo bipartito $G[S, T]$. El objetivo es determinar el mayor tamaño de un corte en un grafo. Este problema es W[1]-difícil parametrizado por clique-width [86] y se puede modelar como un problema 1-localmente verificable con funciones contadoras de colores de la siguiente manera:

- $\text{COLORS} = \{S, T\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{Q}^{-\infty}, \geq, +)$,
- $w'(v, S, \mu) = \frac{\mu_T}{2}$ y $w'(v, T, \mu) = \frac{\mu_S}{2}$, y
- $check'(v, a, \mu) = \text{TRUE}$.

En otras palabras, todo posible coloreo es propio y su peso es precisamente la cantidad de aristas cuyos extremos son de distinto color (son las aristas del corte). Entonces, un coloreo de mínimo peso (según el orden $\preceq$) se corresponde efectivamente con un corte máximo. En este modelo, la cantidad de colores es constante, la función de verificación es 1-estable y 2-modular, y la función de peso es contadora de colores pero no es d -estable ni m -modular para ninguna constante d, m .

La Tabla 4.1 resume estos resultados. Además de los casos indicados con “?”, todavía quedan algunas otras preguntas abiertas:

- ¿Existe un problema 1-localmente verificable con funciones contadoras de colores y cantidad logarítmica de colores, que sea para-NP-difícil parametrizado por clique-width?
- ¿Existe un problema 1-localmente verificable con funciones d -estables o m -modulares con cantidad logarítmica de colores, que sea W[1]-difícil parametrizado por clique-width?

Restricciones sobre <i>check</i> y <i>w</i>	Restricciones sobre $ \text{COLORS} $		
	Constante	Polinomial	No acotada
d -estable	FPT ⁽¹⁾	para-NP-h	para-NP-h
m -modular	FPT ⁽¹⁾	para-NP-h	para-NP-h
Contadora de colores	XP, W[1]-h	para-NP-h	para-NP-h
Sistema de vecindades parciales universal constante	FPT ⁽¹⁾	FPT	?
Sistema de vecindades parciales constante	para-NP-h	para-NP-h	para-NP-h
Sistema de vecindades parciales universal polinomial	XP, W[1]-h	XP, W[1]-h	?
Sistema de vecindades parciales polinomial	para-NP-h	para-NP-h	para-NP-h

Tabla 4.1: Resumen de los resultados principales sobre la complejidad parametrizada por clique-width de problemas 1-localmente verificables sin etiquetas en las aristas, bajo distintas restricciones sobre la cantidad de colores y las funciones de verificación y peso. (1) FPT lineal si una expresión clique-width adecuada viene dada. Aquí, cuando indicamos para-NP-h (o W[1]-h), estamos diciendo que existe un problema con esas condiciones que es para-NP-h (o W[1]-h) parametrizado por clique-width.

Capítulo 5

Parametrización por mim-width

En este capítulo, continuamos con el estudio de problemas localmente verificables pero enfocándonos en grafos de mim-width acotada. Damos un algoritmo que prueba que los problemas que tienen cantidad constante de colores, funciones d -estables a t distancias, para algunas constantes d y t , y propiedades globales de tamaño, conectividad y de composición de condiciones locales, son XP parametrizados por la mim-width de un árbol binario de descomposición dado. Más adelante, en la Sección 8.3, veremos cómo resolver estos problemas cuando además incluyen propiedades globales de aciclicidad. Por otro lado, mostramos ejemplos de problemas localmente verificables que no cumplen alguna de estas condiciones y son para-NP-difíciles.

5.1. Algoritmo

Veremos un algoritmo para resolver problemas 1-localmente verificables con funciones d -estables y con propiedades globales de tamaño y conectividad. Más adelante veremos cómo extenderlo con propiedades globales de composición de condiciones locales. El algoritmo principal que presentaremos está basado en aquel dado por Bui-Xuan, Telle y Vatschelle en [50] e incluye también las ideas de Bergougnoux y Kanté en [27] para propiedades globales de conectividad. Antes de comenzar con la explicación del algoritmo, debemos definir algunos términos y notación que utilizaremos a lo largo de este capítulo.

Sea d un entero positivo. Definimos una relación de equivalencia $=_d$ de modo que dos enteros no-negativos i y j son equivalentes si y sólo si $\min(d, i) = \min(d, j)$. Dado un árbol binario de descomposición (T, δ) de un grafo G y un nodo $v \in V(T)$, hacemos uso de la relación de equivalencia $\equiv_d^v$ presentada en [50]. Esta relación se define sobre subconjuntos de T_v de modo que dos subconjuntos $S, S' \subseteq T_v$ son equivalentes si y sólo si para cualquier $y \in T_{\bar{v}}$ tenemos $|N(y) \cap S| =_d |N(y) \cap S'|$. En base a $\equiv_d^v$, definimos la relación de equivalencia $\equiv_{d,q}^v$ sobre q -tuplas de subconjuntos de T_v , en la cual dos q -tuplas $X, X' \in (\mathcal{P}(T_v))^q$ son equivalentes si y sólo si $X_i \equiv_d^v X'_i$ para todo $i \in [1, q]$ (es decir, si son equivalentes según $\equiv_d^v$ en cada una de sus entradas). Análogamente, definimos las relaciones de equivalencia $\equiv_d^{\bar{v}}$ y $\equiv_{d,q}^{\bar{v}}$ en las cuales dos subconjuntos S, S' de $T_{\bar{v}}$ son equivalentes si y sólo si para cada vértice w en T_v tenemos $|N(w) \cap S| =_d |N(w) \cap S'|$, y análogamente para tuplas en $(\mathcal{P}(T_{\bar{v}}))^q$. Usaremos estas relaciones principalmente con particiones de T_v y no con tuplas arbitrarias de subconjuntos de T_v .

Para una relación de equivalencia $\equiv$ sobre un conjunto S y dado $X \in S$, notamos

$[X]^\equiv$ a la clase de equivalencia que consiste de todos los elementos $X' \in S$ tales que $X' \equiv X$. Cuando quede claro por contexto cuál es la relación de equivalencia, la omitiremos y simplemente escribiremos $[X]$. Notamos $\mathcal{R}_d^v$ y $\mathcal{R}_{d,q}^v$ a los conjuntos de todas las clases de equivalencia de las relaciones $\equiv_d^v$ y $\equiv_{d,q}^v$, respectivamente, y análogamente para $\bar{v}$. Si $R \in \mathcal{R}_{d,q}^v$ y $X \in (\mathcal{P}(T_v))^q$, escribimos $X \equiv_{d,q}^v R$ si X pertenece a la clase de equivalencia R y decimos que X es un *representante de R* (y análogamente para $\bar{v}$). Si $R \in \mathcal{R}_d^v$ y $w \in T_{\bar{v}}$, definimos $|N(w) \cap R|$ como el único número $i \in \llbracket 0, d \rrbracket$ tal que $i =_d |N(w) \cap S|$ para algún $S \subseteq T_v$ con $S \equiv_d^v R$. Esto está bien definido por la definición de $\equiv_d^v$. Decimos que w es *adyacente a R* si $|N(w) \cap R| \geq 1$ y decimos que un conjunto $S \subseteq T_{\bar{v}}$ es *adyacente a R* si contiene un vértice adyacente a R . Si $C \subseteq \llbracket 1, q \rrbracket$ y $X \in (\mathcal{P}(T_v))^q$, notamos X_C al conjunto $\bigcup_{i \in C} X_i \subseteq T_v$ y también definimos la clase de equivalencia $[X]_C$ de la relación $\equiv_d^v$ como $[X]_C = [X_C]$. Definimos también todo esto análogamente para $T_{\bar{v}}$. Definimos $\text{snec}_d(T) = \max_{v \in V(T)} (\max\{|\mathcal{R}_d^v|, |\mathcal{R}_d^{\bar{v}}|\})$ y $\text{snec}_{d,q}(T) = \max_{v \in V(T)} (\max\{|\mathcal{R}_{d,q}^v|, |\mathcal{R}_{d,q}^{\bar{v}}|\})$.

Observación 5.1.1. En [50, Lema 1] se mostró que, para un nodo v de T , podemos computar una lista que contenga exactamente un representante de cada clase de equivalencia de $\equiv_d^v$ en tiempo $O(\text{snec}_d(T) \log(\text{snec}_d(T)) |V(G)|^2)$. Lo mismo aplica a $\equiv_d^{\bar{v}}$. Por [23, Lema 2], $\text{snec}_d(T) \leq n^{d \cdot \text{mimw}(T, \delta)}$. Como las clases de equivalencia de $\equiv_{d,q}^v$ y $\equiv_{d,q}^{\bar{v}}$ son q -tuplas de clases de equivalencia de $\equiv_d^v$ y $\equiv_d^{\bar{v}}$, respectivamente, sus representantes son simplemente las q -tuplas de los representantes de $\equiv_d^v$ y $\equiv_d^{\bar{v}}$, respectivamente. Luego, también podemos calcular eficientemente una lista de representantes y además $\text{snec}_{d,q}(T) \leq n^{q \cdot d \cdot \text{mimw}(T, \delta)}$.

El siguiente lema muestra que todo problema 1-localmente verificable con cantidad constante de colores, funciones d -estables y propiedades globales de conectividad positiva dadas por un conjunto $\mathcal{C} \subseteq \mathcal{P}(\llbracket 1, q \rrbracket)$ y propiedades globales de tamaño dadas por un conjunto $\mathcal{K}$ con $|\mathcal{K}| = 1$, es XP parametrizado por la mim-width de un árbol binario de descomposición dado. Más adelante explicaremos cómo este resultado puede extenderse fácilmente a un conjunto $\mathcal{K}$ con más de un elemento, a propiedades globales de conectividad negativa, y a funciones de verificación y peso d -estables a t distancias. Para el análisis de complejidad, utilizaremos el hecho de que d y q son constantes y que $|\mathcal{C}| \leq 2^q$ (y por lo tanto está acotado por una constante). Esto además implica que las funciones $check'$ y w' asociadas a $check$ y w , respectivamente, cuyos tiempos dependen únicamente de d y q , son computables en tiempo constante. Supondremos que $L_v = \text{COLORS}$ para todo vértice v , y notemos que esto no supone una restricción adicional (si esto no ocurriese, podríamos modificar la función de verificación para incluir esta restricción del color allí).

Lema 5.1.2 ([103, Lema 5.2]). *Sean d y q enteros positivos, con $q \geq 2$. Consideremos un problema 1-localmente verificable Π con q colores, funciones de verificación y peso d -estables, propiedades globales de tamaño dadas por un conjunto $\mathcal{K}$ que contiene un único elemento K , y propiedades globales de conectividad positiva dadas por un conjunto $\mathcal{C}$. Luego, para un grafo de entrada G con un árbol binario de descomposición (T, δ) asociado, podemos resolver Π en tiempo $O(n^{2q} \cdot \text{snec}_{d,q}(T)^{6|\mathcal{C}|+6} \cdot (n^2 + \text{snec}_{d,q}(T)^{\omega-1}))$, donde $n = |V(G)|$ y ω es el exponente de multiplicación de matrices ($\omega < 2,4$ [35]).*

Demostración. Esta demostración sigue de cerca a las dadas en [50] y [27]. Nuestra contribución es la generalización de estas ideas a cualquier función de verificación d -

estable, incluyendo además la función de peso y complementando con las propiedades globales.

Sean $check'$ y w' funciones asociadas a $check$ y w , respectivamente, las cuales estamos suponiendo que son d -estables. Primero que nada, notemos que encontrar un coloreo c de G con q colores $\{a_1, \dots, a_q\}$ es equivalente a encontrar una q -partición $(X_1, \dots, X_q)$ de $V(G)$, donde $X_j = c^{-1}(a_j)$ para todo $j \in \llbracket 1, q \rrbracket$. Por lo tanto vamos a mostrar cómo calcular una q -partición $X \in \text{part}_q(V(G))$ tal que $check'(v, a, (|N(v) \cap X_1|, \dots, |N(v) \cap X_q|)) = \text{TRUE}$ para todo $v \in V(G)$ y $a \in \text{COLORS}$, $|X| = K$ y $G[X_C]$ es conexo para todo $C \in \mathcal{C}$. Supondremos que G no tiene vértices aislados (si así lo hiciera, podríamos resolver el problema independientemente en cada componente conexa).

Para cada $v \in V(T)$, $X \in \text{part}_q(T_v)$, $R' \in \mathcal{R}_{d,q}^{\bar{v}}$ y cada $Y \in \text{part}_q(T_{\bar{v}})$ con $Y \equiv_{d,q}^{\bar{v}} R'$, definimos el *peso de X cuando se complementa con Y* , $w(X, Y)$, como

$$w(X, Y) = \bigoplus_{j \in \llbracket 1, q \rrbracket} \bigoplus_{w \in X_j} w'(w, a_j, (|N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|)).$$

Y definimos $w(X, [Y]) = w(X, Y)$. Notemos que, como w es d -estable y por definición de la relación de equivalencia $\equiv_{d,q}^{\bar{v}}$, esta noción está bien definida.

Decimos que una tupla $(X, Y) \in \text{part}_q(T_v) \times \text{part}_q(T_{\bar{v}})$ es T_v -válida si para cualquier vértice $w \in T_v$ tenemos $check'(w, j_w, (|N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|)) = \text{TRUE}$, donde j_w es tal que $w \in X_{j_w}$, $|X| + |Y| = K$ y $G[X_C \cup Y_C]$ es conexo para cada $C \in \mathcal{C}$. Decimos que una tupla $(X, R') \in \text{part}_q(T_v) \times \mathcal{R}_{d,q}^{\bar{v}}$ es T_v -parcialmente válida si para cualquier vértice $w \in T_v$ y para un representante $Y \equiv_{d,q}^{\bar{v}} R'$ tenemos $check'(w, j_w, (|N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|)) = \text{TRUE}$, donde j_w es tal que $w \in X_{j_w}$. Observemos que todas estas nociones están bien definidas por la definición de la relación de equivalencia $\equiv_{d,q}^{\bar{v}}$.

Explicaremos ahora la idea general del algoritmo. Para cada nodo $v \in V(T)$ y para cada $R \in \mathcal{R}_{d,q}^v$, $R' \in \mathcal{R}_{d,q}^{\bar{v}}$, $K' \in \text{part}_q(|T_v|)$, computaremos un conjunto $M_v[R, R', K'] \subseteq \text{part}_q(T_v)$ tal que

- $|M_v[R, R', K']| \leq 2^{|\mathcal{C}|} \text{snec}_{d,q}(T)^{2^{|\mathcal{C}|}}$,
- para cada $X \in M_v(R, R', K')$ tenemos que $X \equiv_{d,q}^v R$, $|X| = K'$ y que (X, R') es T_v -parcialmente válida, y
- para cada $X \in \text{part}_q(T_v)$ con $X \equiv_{d,q}^v R$, $|X| = K'$ y cada $Y \in \text{part}_q(T_{\bar{v}})$ tal que $Y \equiv_{d,q}^{\bar{v}} R'$ y (X, Y) es T_v -válida existe un $X' \in M_v[R, R', K']$ tal que $w(X', Y) \preceq w(X, Y)$ y (X', Y) es T_v -válida.

Sea r la raíz de T . Observemos que $T_{\bar{r}} = \emptyset$ y entonces para cada $X, X' \in \text{part}_q(T_r)$, tenemos que $X \equiv_{d,q}^r X'$. Además tenemos que $\mathcal{R}_{d,q}^r$ contiene sólo un elemento: $[\emptyset^q]$. Sigue que $M_r[[\emptyset^q], [\emptyset^q], K]$ tiene que contener una solución óptima a nuestro problema, si y sólo si tal solución existe. Entonces para resolver el problema sólo tenemos que buscar un $X \in M_r[[\emptyset^q], [\emptyset^q], K]$ de mínimo peso $w(X, \emptyset^q)$ tal que $(X, \emptyset^q)$ es T_r -válida. Notemos que el tamaño de $M_r[[\emptyset^q], [\emptyset^q], K]$ es a lo sumo $2^{|\mathcal{C}|} \text{snec}_{d,q}(T)^{2^{|\mathcal{C}|}}$, y que constatar si $(X, \emptyset^q)$ es T_r -válida (sabiendo que $(X, [\emptyset^q])$ es T_r -parcialmente válida) y calcular $w(X, \emptyset^q)$ se pueden hacer en tiempo $O(n^2)$. Por lo tanto, dado el conjunto $M_r[[\emptyset^q], [\emptyset^q], K]$, encontrar una solución óptima allí (o descubrir que no existe ninguna solución) puede hacerse en tiempo $O(\text{snec}_{d,q}(T)^{2^{|\mathcal{C}|}} n^2)$.

Vamos a calcular estos conjuntos $M_v[R, R', K']$ empleando técnicas estándar de programación dinámica, en el estilo bottom-up. Comenzaremos por el caso en el que v es una hoja de T . El conjunto T_v contiene un único vértice w de G y para cada clase de equivalencia en $\mathcal{R}_{d,q}^v$ existe un único representante. Así, para cada $R \in \mathcal{R}_{d,q}^v$ y $R' \in \mathcal{R}_{d,q}^{\bar{v}}$, llamamos X al único representante de R y definimos $M_v[R, R', |X|]$ como $\{X\}$ si $X \in \text{part}_q(T_v)$ y (X, R') es T_v -parcialmente válida, y como $\emptyset$ si no. Para cada $K' \neq |X|$, definimos $M_v[R, R', K'] = \emptyset$.

Ahora sea v un nodo interno de T (un nodo que no es una hoja). Sean a y b los hijos de v , y supongamos que ya calculamos correctamente los conjuntos $M_a[R_a, R'_a, K_a]$ para cada $R_a \in \mathcal{R}_{d,q}^a$, $R'_a \in \mathcal{R}_{d,q}^{\bar{a}}$ y $K_a \in \text{part}_q(|T_a|)$, así como $M_b[R_b, R'_b, K_b]$ para cada $R_b \in \mathcal{R}_{d,q}^b$, $R'_b \in \mathcal{R}_{d,q}^{\bar{b}}$ y $K_b \in \text{part}_q(|T_b|)$. Para $R \in \mathcal{R}_{d,q}^v$, $R' \in \mathcal{R}_{d,q}^{\bar{v}}$ y $K' \in \text{part}_q(|T_v|)$, decimos que dos tuplas $(R_a, R'_a, K^a) \in \mathcal{R}_{d,q}^a \times \mathcal{R}_{d,q}^{\bar{a}} \times \text{part}_q(|T_a|)$ y $(R_b, R'_b, K^b) \in \mathcal{R}_{d,q}^b \times \mathcal{R}_{d,q}^{\bar{b}} \times \text{part}_q(|T_b|)$ son (R, R', K') -compatibles si $K^a + K^b = K'$ y, para representantes $Y, Y', Y_a, Y'_a, Y_b, Y'_b$ de $R, R', R_a, R'_a, R_b, R'_b$ respectivamente, tenemos $Y_a \cup Y_b \equiv_{d,q}^v Y$, $Y_b \cup Y' \equiv_{d,q}^{\bar{a}} Y'_a$, $Y_a \cup Y' \equiv_{d,q}^{\bar{b}} Y'_b$. Para algunos $R \in \mathcal{R}_{d,q}^v$, $R' \in \mathcal{R}_{d,q}^{\bar{v}}$ y $K' \in \text{part}_q(|T_v|)$ fijos consideramos el conjunto

$$\mathcal{M} = \bigcup_{\substack{(R_a, R'_a, K^a) \text{ y } (R_b, R'_b, K^b) \\ \text{son } (R, R', K')\text{-compatibles}}} \left(\bigcup_{\substack{X_a \in M_a[R_a, R'_a, K^a] \\ X_b \in M_b[R_b, R'_b, K^b]}} (X_a \cup X_b) \right).$$

Primero mostraremos que este conjunto tiene todas las propiedades esperadas para $M_v[R, R', K']$ excepto por la cota superior del tamaño, y luego mostraremos una forma de encoger este conjunto sin perder dichas propiedades. Notemos que el tamaño de $\mathcal{M}$ puede llegar a ser $n^q \cdot \text{snec}_{d,q}(T)^4 \cdot (2^{|C|} \text{snec}_{d,q}(T)^{2|C|})^2$ porque estamos suponiendo que $M_a[R_a, R'_a, K^a]$ y $M_b[R_b, R'_b, K^b]$ tienen tamaño a lo sumo $2^{|C|} \text{snec}_{d,q}(T)^{2|C|}$ y existen a lo sumo $n^q \cdot \text{snec}_{d,q}(T)^4$ pares de tuplas (R_a, R'_a, K^a) y (R_b, R'_b, K^b) que son (R, R', K') -compatibles (hay a lo sumo n^q posibles valores para K^a y para cada uno de ellos existe una sola opción para K^b , y existen a lo sumo $\text{snec}_{d,q}(T)$ posibles valores para cada R_a, R'_a, R_b, R'_b).

Sea (R, R', K') como mencionamos arriba y sea $X \in \text{part}_q(T_v)$ tal que $X \equiv_{d,q}^v R$ y $|X| = K'$. Supongamos que existe $Y \in \text{part}_q(T_{\bar{v}})$ tal que $Y \equiv_{d,q}^{\bar{v}} R'$ y (X, Y) es T_v -válida. Sean $X^a = T_a \cap X$, $X^b = T_b \cap X$, $X'^a = X^b \cup Y$, $X'^b = X^a \cup Y$, $K^a = |X^a|$ y $K^b = |X^b|$. Claramente (X^a, X'^a) es T_a -válida, (X^b, X'^b) es T_b -válida y $K^a + K^b = K'$, entonces $M_a[[X^a], [X'^a], K^a]$ y $M_b[[X^b], [X'^b], K^b]$ son no-vacíos.

Notamos $\text{CONN}(Z^a, Z^b, Y, C)$ al enunciado “ $(Z^a)_C \cup (Z^b)_C \cup Y_C$ es conexo para cada $C \in \mathcal{C}$ ”. Observemos que

$$\begin{aligned} w(X, Y) &= w(X^a \cup X^b, Y) \\ &= w(X^a, X'^a) \oplus w(X^b, X'^b) \\ &\succeq \min_{\substack{Z^a \in \text{part}_q(T_a), Z^a \equiv_{d,q}^a X^a \\ Z^b \in \text{part}_q(T_b), Z^b \equiv_{d,q}^b X^b \\ |Z^a| = K^a, |Z^b| = K^b \\ \text{CONN}(Z^a, Z^b, Y, C)}} (w(Z^a, X'^a) \oplus w(Z^b, X'^b)) \end{aligned}$$

$$\begin{aligned}
&= \min_{\substack{Z^a \in \text{part}_q(T_a), Z^a \equiv_{d,q}^a X^a \\ |Z^a| = K^a}} \left(w(Z^a, X'^a) \oplus \left(\min_{\substack{Z^b \in \text{part}_q(T_b), Z^b \equiv_{d,q}^b X^b \\ |Z^b| = K^b \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} w(Z^b, X'^b) \right) \right) \\
&= \min_{\substack{Z^a \in \text{part}_q(T_a), Z^a \equiv_{d,q}^a X^a \\ |Z^a| = K^a}} \left(w(Z^a, X'^a) \oplus \left(\min_{\substack{Z^b \in M_b[[X^b], [X'^b], K^b] \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} w(Z^b, X'^b) \right) \right) \\
&= \min_{\substack{Z^a \in \text{part}_q(T_a), Z^a \equiv_{d,q}^a X^a \\ |Z^a| = K^a \\ Z^b \in M_b[[X^b], [X'^b], K^b] \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} (w(Z^a, X'^a) \oplus w(Z^b, X'^b)) \\
&= \min_{Z^b \in M_b[[X^b], [X'^b], K^b]} \left(\left(\min_{\substack{Z^a \in \text{part}_q(T_a), Z^a \equiv_{d,q}^a X^a \\ |Z^a| = K^a \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} w(Z^a, X'^a) \right) \oplus w(Z^b, X'^b) \right) \\
&= \min_{Z^b \in M_b[[X^b], [X'^b], K^b]} \left(\left(\min_{\substack{Z^a \in M_a[[X^a], [X'^a], K^a] \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} w(Z^a, X'^a) \right) \oplus w(Z^b, X'^b) \right) \\
&= \min_{\substack{Z^a \in M_a[[X^a], [X'^a], K^a] \\ Z^b \in M_b[[X^b], [X'^b], K^b] \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} (w(Z^a, X'^a) \oplus w(Z^b, X'^b)) \\
&= \min_{\substack{Z^a \in M_a[[X^a], [X'^a], K^a] \\ Z^b \in M_b[[X^b], [X'^b], K^b] \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} (w(Z^a, Z^b \cup Y) \oplus w(Z^b, Z^a \cup Y)) \\
&= \min_{\substack{Z^a \in M_a[[X^a], [X'^a], K^a] \\ Z^b \in M_b[[X^b], [X'^b], K^b] \\ \text{CONN}(Z^a, Z^b, Y, \mathcal{C})}} w(Z^a \cup Z^b, Y).
\end{aligned}$$

Sigue que existen $Z^a \in M_a[[X^a], [X'^a], K^a]$ y $Z^b \in M_b[[X^b], [X'^b], K^b]$ tales que $w(Z^a \cup Z^b, Y) \preceq w(X^a \cup X^b, Y) = w(X, Y)$. Es más, $(Z^a \cup Z^b, Y)$ es T_v -válida. En efecto, las restricciones de conectividad son satisfechas por lo mostrado arriba, y las demás por la definición de M_a y M_b .

En síntesis, para cada $X \in \text{part}_q(T_v)$ con $X \equiv_{d,q}^v R$ y $|X| = K'$, y cada $Y \in \text{part}_q(T_{\bar{v}})$ tal que $Y \equiv_{d,q}^{\bar{v}} R'$ y (X, Y) es T_v -válida, existe $Z \in \mathcal{M}$ tal que $w(Z, Y) \preceq w(X, Y)$ y (Z, Y) es T_v -válida.

Ahora vamos a mostrar que podemos encontrar un subconjunto de $\mathcal{M}$ de cardinalidad a lo sumo $2^{|C|} |\text{snec}_{d,q}(T)|^{2|C|}$ que además mantiene la propiedad que acabamos de mostrar. Observemos que cualquier subconjunto de $\mathcal{M}$ sólo puede contener particiones de T_v que son T_v -parcialmente válidas junto con R' y que tienen el tamaño de partición correcto. Entonces, al elegir un subconjunto de $\mathcal{M}$, sólo nos tenemos que enfocar en mantener las soluciones parciales que forman parte de una solución óptima.

Sean $v \in V(T)$, $R' \in \mathcal{R}_{d,q}^{\bar{v}}$, $\mathcal{X} \subseteq \text{part}_q(T_v)$ y $\mathcal{X}' \subseteq \mathcal{X}$. Decimos que $\mathcal{X}'$ “ (v, R') -representa” a $\mathcal{X}$ si para cada $X \in \mathcal{X}$ e $Y \in \text{part}_q(T_{\bar{v}})$ tales que $Y \equiv_{d,q}^{\bar{v}} R'$ y $X_C \cup Y_C$ es conexo para cada $C \in \mathcal{C}$, existe $X' \in \mathcal{X}'$ tal que $X'_C \cup Y_C$ es conexo para cada $C \in \mathcal{C}$ y

$$w(X', R') \preceq w(X, R').$$

Afirmación 5.1.3. Sean $v \in V(T)$, $R \in \mathcal{R}_{d,q}^v$ y $R' \in \mathcal{R}_{d,q}^{\bar{v}}$. Sea $\mathcal{X} \subseteq \text{part}_q(T_v)$ tal que $X \equiv_{d,q}^v R$ para cada $X \in \mathcal{X}$. Podemos encontrar un conjunto que (v, R') -represente a $\mathcal{X}$ y de cardinalidad a lo sumo $2^{|\mathcal{C}|} \text{snec}_{d,q}(T)^{2|\mathcal{C}|}$ en tiempo $O(|\mathcal{X}| \cdot \text{snec}_{d,q}(T)^{2|\mathcal{C}|} \cdot (n^2 + \text{snec}_{d,q}(T)^{\omega-1}))$, donde ω es el exponente de multiplicación de matrices ($\omega < 2,4$ [35]).

Demostración. Sea $\mathcal{C}_\emptyset \subseteq \mathcal{C}$ el conjunto de restricciones de conectividad C para las cuales $\emptyset \equiv_d^{\bar{v}} R'_C$. Notemos que $Y_C \equiv_d^{\bar{v}} \emptyset$ para cada $C \in \mathcal{C}_\emptyset$ y cada $Y \in \text{part}_q(T_{\bar{v}})$ con $Y \equiv_{d,q}^{\bar{v}} R'$. Además, si $Y_C = \emptyset$ entonces Y sólo puede completarse a una tupla T_v -válida con una partición $X \in \mathcal{X}$ cuando X_C es conexo. Por otro lado, si $Y_C \neq \emptyset$ entonces Y sólo puede completarse a una tupla T_v -válida con una partición $X \in \mathcal{X}$ cuando X_C es vacío, ya que si no $X_C \cup Y_C$ no podría ser conexo.

Dado un mapeo $\tau: \mathcal{C}_\emptyset \rightarrow \{0, 1\}$, decimos que una partición $X \in \mathcal{X}$ es *de tipo* τ si $X_C = \emptyset$ para cada $C \in \tau^{-1}(0)$ y X_C es conexo para cada $C \in \tau^{-1}(1)$. Observemos que si $\mathcal{C}_\emptyset = \emptyset$ entonces τ sólo puede ser el mapeo vacío y entonces cada X es de tipo τ . Existen a lo sumo $2^{|\mathcal{C}_\emptyset|}$ mapeos posibles de $\mathcal{C}_\emptyset$ a $\{0, 1\}$. Para cada tal mapeo τ calcularemos un conjunto $\mathfrak{X}_\tau$ de tamaño a lo sumo $\text{snec}_{d,q}(T)^{2|\mathcal{C}|}$ que (v, R') -represente al conjunto $\{X \in \mathcal{X} : X \text{ es de tipo } \tau\}$. Afirmamos que su unión determina un conjunto como el demandado. En efecto, para cada $Y \equiv_{d,q}^{\bar{v}} R'$ definimos un mapeo τ_Y como arriba que relacione $C \in \mathcal{C}_\emptyset$ con 0 si Y_C es no-vacío y con 1 si es vacío. Por lo explicado anteriormente, para cualquier $X \in \mathcal{X}$ tal que $X_C \cup Y_C$ es conexo para cada $C \in \mathcal{C}$, tenemos que X es de tipo τ_Y . Esto implica que existe $X' \in \mathfrak{X}_{\tau_Y}$ tal que $X'_C \cup Y_C$ es conexo para cada $C \in \mathcal{C}$ y $w(X', Y) \preceq w(X, Y)$. Luego, la unión de todos los $\mathfrak{X}_\tau$ es un (v, R') -representante de $\mathcal{X}$. Más aún, si el tamaño de cada $\mathfrak{X}_\tau$ es a lo sumo $\text{snec}_{d,q}(T)^{2|\mathcal{C}|}$, entonces su unión tiene tamaño a lo sumo $2^{|\mathcal{C}_\emptyset|} \text{snec}_{d,q}(T)^{2|\mathcal{C}|} \leq 2^{|\mathcal{C}|} \text{snec}_{d,q}(T)^{2|\mathcal{C}|}$.

Sólo resta mostrar cómo calcular un tal conjunto $\mathfrak{X}_\tau$ para cada τ . Consideremos entonces un mapeo τ como los que describimos antes y también consideremos el conjunto $\mathcal{X}_\tau$ que consiste de cada $X \in \mathcal{X}$ de tipo τ . Sea $\overline{\mathcal{C}_\emptyset} = \mathcal{C} - \mathcal{C}_\emptyset$. Ahora R'_C no contiene el conjunto vacío para ningún $C \in \overline{\mathcal{C}_\emptyset}$. Sea $\mathcal{Y}$ el conjunto de todas las particiones $Y \equiv_{d,q}^{\bar{v}} R'$ tales que $\tau_Y = \tau$.

Si existe una partición $X \in \mathcal{X}_\tau$ tal que existe un $C \in \overline{\mathcal{C}_\emptyset}$ y una componente conexa $A \subseteq X_C$ tal que A no es adyacente a R'_C , entonces para cualquier $Y \in \mathcal{Y}$ el conjunto $X_C \cup Y_C$ no es conexo. Luego, eliminamos tales particiones de $\mathcal{X}_\tau$. Análogamente, si existe una partición $Y \in \mathcal{Y}$ que tiene una componente conexa A de Y_C tal que A no es adyacente a R_C entonces $X_C \cup Y_C$ no puede ser conexo para ningún $X \in \mathcal{X}_\tau$. Así, también eliminamos estas particiones de $\mathcal{Y}$.

Si el tamaño de $\mathcal{X}_\tau$ es a lo sumo $\text{snec}_{d,q}(T)^{2|\mathcal{C}|}$, entonces podemos establecer $\mathfrak{X}_\tau = \mathcal{X}_\tau$. Luego, podemos suponer de ahora en más que $|\mathcal{X}_\tau| > \text{snec}_{d,q}(T)^{2|\mathcal{C}|}$.

Para cada $S \subseteq T_{\bar{v}}$, sea v_S un vértice en S . Sea $\mathcal{T}$ el conjunto de las $\overline{\mathcal{C}_\emptyset}$ -tuplas cuyas entradas son 2-tuplas de $\mathcal{R}_{d,q}^{\bar{v}}$ (es decir, $\mathcal{T}$ es el conjunto de todos los elementos $(R_C^1, R_C^2)_{C \in \overline{\mathcal{C}_\emptyset}}$ con $R^1, R^2 \in \mathcal{R}_{d,q}^{\bar{v}}$). Definimos ahora las siguientes matrices sobre $\mathbb{F}_2$:

- J es la matriz de $\mathcal{X}_\tau \times \mathcal{Y}$ para la cual $J(X, Y) = 1$ si y sólo si $X_C \cup Y_C$ es conexo para todo $C \in \overline{\mathcal{C}_\emptyset}$.
- L es una matriz de $\mathcal{P}(T_v) \times (\mathcal{R}_d^{\bar{v}} \times \mathcal{R}_d^{\bar{v}})$ tal que $L(X, (R'_1, R'_2)) = 1$ si y sólo si ninguna componente conexa de X es adyacente tanto a R'_1 como a R'_2 .

- $\bar{L}$ es una matriz de $(\mathcal{R}_d^{\bar{v}} \times \mathcal{R}_d^{\bar{v}}) \times \mathcal{P}(T_{\bar{v}})$ tal que $\bar{L}((R'_1, R'_2), Y) = 1$ si y sólo si existe un corte conexo (Y_1, Y_2) de Y tal que $v_Y \in Y_1$, $Y_1 \equiv_d^{\bar{v}} R'_1$ y $Y_2 \equiv_d^{\bar{v}} R'_2$.
- L^* es una matriz de $\mathcal{X}_\tau \times \mathcal{T}$ tal que $L^*(X, (R_C^1, R_C^2)_{C \in \mathcal{C}}) = 1$ si y sólo si $L(X_C, (R_C^1, R_C^2)) = 1$ para cada $C \in \bar{\mathcal{C}}_\emptyset$.
- $\bar{L}^*$ es una matriz de $\mathcal{T} \times \mathcal{Y}$ tal que $\bar{L}^*((R_C^1, R_C^2)_{C \in \mathcal{C}}, Y) = 1$ si y sólo si $\bar{L}((R_C^1, R_C^2), Y_C) = 1$ para cada $C \in \bar{\mathcal{C}}_\emptyset$.

Probaremos que $L^* \bar{L}^* = J$. Todas las operaciones que siguen a continuación son sobre $\mathbb{F}_2$. Para todo $X \in \mathcal{X}_\tau$ y todo $Y \in \mathcal{Y}$, tenemos:

$$\begin{aligned}
L^* \bar{L}^*(X, Y) &= \sum_{(R_C^1, R_C^2)_{C \in \bar{\mathcal{C}}_\emptyset} \in \mathcal{T}} \mathbb{1}(\forall C \in \bar{\mathcal{C}}_\emptyset. L(X_C, (R_C^1, R_C^2)) = 1 \wedge \bar{L}((R_C^1, R_C^2), Y_C) = 1) \\
&= |\{(R_C^1, R_C^2)_{C \in \bar{\mathcal{C}}_\emptyset} \in \mathcal{T} : L(X_C, (R_C^1, R_C^2)) = 1 \wedge \bar{L}((R_C^1, R_C^2), Y_C) = 1 \\
&\quad \text{para cada } C \in \bar{\mathcal{C}}_\emptyset\}| \\
&= |\{(R_C^1, R_C^2)_{C \in \bar{\mathcal{C}}_\emptyset} \in \mathcal{T} : \forall C \in \bar{\mathcal{C}}_\emptyset. \exists (Z_C^1, Z_C^2) \in ccut(X_C \cup Y_C). v_{Y_C} \in Z_C^1 \wedge \\
&\quad Z_C^1 \cap Y_C \equiv_d^{\bar{v}} R_C^1 \wedge Z_C^2 \cap Y_C \equiv_d^{\bar{v}} R_C^2\}| \\
&= \prod_{C \in \bar{\mathcal{C}}_\emptyset} |\{(R^1, R^2) \in \mathcal{R}_d^{\bar{v}} \times \mathcal{R}_d^{\bar{v}} : \exists (Z^1, Z^2) \in ccut(X_C \cup Y_C). v_{Y_C} \in Z^1 \wedge \\
&\quad Z^1 \cap Y_C \equiv_d^{\bar{v}} R^1 \wedge Z^2 \cap Y_C \equiv_d^{\bar{v}} R^2\}| \\
&= \prod_{C \in \bar{\mathcal{C}}_\emptyset} |\{(Z^1, Z^2) \in ccut(X_C \cup Y_C) : v_{Y_C} \in Z^1\}| \\
&= \prod_{C \in \bar{\mathcal{C}}_\emptyset} 2^{|cc(X_C \cup Y_C)|-1}.
\end{aligned}$$

Las primeras dos igualdades valen directamente por la definición de las matrices, y la tercera por [27, Afirmación 4.3.1]. La cuarta vale porque podemos elegir el par de clases de equivalencia (R_C^1, R_C^2) de forma independiente para cada $C \in \bar{\mathcal{C}}_\emptyset$. La quinta igualdad viene dada por [27, Afirmación 4.3.2], y la última por el hecho de que cualquier componente conexas de $X_C \cup Y_C$ que no contiene a v_{Y_C} puede ser asignada arbitrariamente tanto a Z^1 como a Z^2 . Como definimos las matrices sobre $\mathbb{F}_2$, y el único caso en el que cada factor del último producto es impar es cuando $X_C \cup Y_C$ es conexo para cada $C \in \bar{\mathcal{C}}_\emptyset$, nuestra afirmación es efectivamente cierta.

Observemos que podemos computar la matriz L^* en tiempo $O(|\mathcal{X}_\tau| \cdot \text{sne}_{d,q}(T)^{2|C|} \cdot n^2)$. En efecto, la cantidad de entradas de L^* es a lo sumo $|\mathcal{X}_\tau| \cdot \text{sne}_{d,q}(T)^{2|C|}$. Además, para una entrada indexada por $(X, (R_C^1, R_C^2)_{C \in \mathcal{C}})$ y para cada $C \in \bar{\mathcal{C}}_\emptyset$, podemos computar primero las componentes conexas de X_C y luego verificar las adyacencias. Todos estos pasos pueden hacerse en tiempo $O(n^2)$.

Ahora, podemos asignar un peso a cada fila de L^* . Más precisamente, la fila indexada por $X \in \mathcal{X}_\tau$ obtiene el peso $w(X, R')$. Estas asignaciones se pueden hacer en tiempo $O(|\mathcal{X}_\tau| n^2)$. Sea $\mathcal{B} \subseteq \mathcal{X}_\tau$ tal que las filas de L^* que están indexadas por las entradas en $\mathcal{B}$ son una base del espacio fila de L^* , $\text{row}(L^*)$, y la suma de los pesos de estas filas es mínimo entre todas las bases de $\text{row}(L^*)$ que consisten de los vectores fila de L^* . Observemos que la dimensión de $\text{row}(L^*)$, y entonces también el tamaño de $\mathcal{B}$,

es a lo sumo la cantidad de columnas de L^* , la cual está acotada superiormente por $\text{sne}_{d,q}(T)^{2|C|}$. Notemos también que, por [35, Lema 3.15], $\mathcal{B}$ puede ser calculado en tiempo $O(|\mathcal{X}_\tau| \cdot \text{sne}_{d,q}(T)^{2|C|(\omega-1)})$ (donde $\omega < 2,4$ es el exponente de multiplicación de matrices).

Afirmamos que $\mathcal{B}$ es un conjunto que tiene las propiedades demandadas por $\mathfrak{X}_\tau$. El tamaño ya es correcto, así que resta mostrar que para cada $X \in \mathcal{X}_\tau$, $Y \in \mathcal{Y}$, con $J(X, Y) = 1$, existe un $X' \in \mathcal{B}$ tal que $J(X', Y) = 1$ y $w(X', Y) \preceq w(X, Y)$. En efecto, como $\mathcal{B}$ corresponde a una base sobre $\mathbb{F}_2$, existe un único $\mathcal{B}' \subseteq \mathcal{B}$ tal que para cualquier $\rho \in \mathcal{T}$ tenemos

$$L^*(X, \rho) = \sum_{B \in \mathcal{B}'} L^*(B, \rho).$$

Podemos usar esto para reescribir la fila de J correspondiente a X :

$$\begin{aligned} J(X, Y) &= \sum_{\rho \in \mathcal{T}} L^*(X, \rho) \overline{L^*}(\rho, Y) \\ &= \sum_{B \in \mathcal{B}'} \sum_{\rho \in \mathcal{T}} L^*(B, \rho) \overline{L^*}(\rho, Y) \\ &= \sum_{B \in \mathcal{B}'} J(B, Y). \end{aligned}$$

De esta manera, si $J(X, Y) = 1$ entonces existe una cantidad impar de elementos $B \in \mathcal{B}'$ tales que $J(B, Y) = 1$. Sea B cualquiera de ellos de peso máximo. Si $w(X, Y) \prec w(B, Y)$ entonces $\mathcal{B} - \{B\} \cup \{X\}$ corresponde a una base del espacio fila de L^* que tiene menor peso que el correspondiente a $\mathcal{B}$. En efecto, es claro que el peso es más chico por la suposición sobre su peso y que efectivamente corresponde a una base, ya que $L^*(B, \rho) = L^*(X, \rho) - \sum_{B' \in \mathcal{B} - \{B\}} L^*(B', \rho)$ para cada $\rho \in \mathcal{T}$. $\lrcorner$

Directo de la Afirmación 5.1.3 sigue que existe un conjunto de tamaño a lo sumo $2^{|C|} \text{sne}_{d,q}(T)^{2|C|}$ que (v, R') -representa a $\mathcal{M}$. Por lo tanto podemos definir $M_v[R, R', K']$ como cualquiera de estos conjuntos. Sólo resta calcular el tiempo de ejecución de todo este algoritmo.

Podemos precalcular un representante para cada clase de equivalencia usando lo mencionado en la Observación 5.1.1. Para un nodo fijo $v \in V(T)$, la matriz M_v tiene a lo sumo $\text{sne}_{d,q}(T)^2 \cdot n^{q-1}$ entradas. Calculamos cada una de estas entradas mediante programación dinámica de abajo hacia arriba (bottom-up). Cuando v es una hoja de T , cada una de esas entradas se puede calcular en tiempo constante. En el otro caso (cuando v es un nodo interno), primero calculamos $\mathcal{M}$ en tiempo $O(n^q \cdot \text{sne}_{d,q}(T)^4 \cdot (2^{|C|} \cdot \text{sne}_{d,q}(T)^{2|C|})^2 \cdot n^2)$ usando los conjuntos precalculados sobre a y b (para cada uno de los $O(n^q \cdot \text{sne}_{d,q}(T)^4)$ pares de tuplas compatibles, iteramos sobre todos los posibles X_a, X_b , que son a lo sumo $(2^{|C|} \cdot \text{sne}_{d,q}(T)^{2|C|})^2$ posibilidades, y en tiempo $O(n^2)$ computamos $X_a \cup X_b$), y luego aplicamos la Afirmación 5.1.3 en tiempo $O(n^q \cdot \text{sne}_{d,q}(T)^4 \cdot (2^{|C|} \cdot \text{sne}_{d,q}(T)^{2|C|})^2 \cdot \text{sne}_{d,q}(T)^{2|C|} \cdot (n^2 + \text{sne}_{d,q}(T)^{\omega-1}))$, ya que en este caso tenemos $|\mathcal{X}| = |\mathcal{M}| = n^q \cdot \text{sne}_{d,q}(T)^4 \cdot (2^{|C|} \cdot \text{sne}_{d,q}(T)^{2|C|})^2$. Sigue que cada entrada de M_v puede calcularse en tiempo $O(n^q \cdot \text{sne}_{d,q}(T)^{6|C|+4} \cdot (n^2 + \text{sne}_{d,q}(T)^{\omega-1}))$. Luego sólo necesitamos multiplicar por el número de posibles nodos $v \in V(T)$, que es $O(n)$, y por el número de posibles entradas de M_v , que es $O(\text{sne}_{d,q}(T)^2 \cdot n^{q-1})$. Concluimos sumando la complejidad del análisis en la raíz. Así, obtenemos el tiempo total expresado en el enunciado. $\square$

En esta demostración consideramos solamente el caso en el que queremos que el tamaño de cada clase de color sea igual a un cierto número. No obstante, esto es suficiente para manejar *cualquier* propiedad global de tamaño. En efecto, si la propiedad global de tamaño requiere que el tamaño de las q clases de colores esté contenido en algún conjunto $\mathcal{K} \subseteq \text{part}_q(n)$, siendo n la cantidad de vértices del grafo de entrada, entonces podemos calcular la solución óptima para cada $K \in \mathcal{K}$ utilizando el algoritmo anterior, y buscar la mejor entre todas ellas. Como $|\mathcal{K}| \leq n^{q-1}$, estamos aplicando el algoritmo no más que esta cantidad de veces.

Para poder trabajar con propiedades globales de conectividad negativa dadas por un conjunto $\mathcal{C}^-$, primero notemos que pedir que un conjunto S no sea conexo es equivalente a pedir que S se pueda partir en dos subconjuntos no vacíos S_0 y S_1 tales que no haya aristas entre vértices en S_0 y vértices en S_1 . Entonces podemos hacer lo siguiente. Reemplazamos el conjunto de colores COLORS con el conjunto $\text{COLORS} \times (\prod_{C \in \mathcal{C}^-} \{0, 1\})$. Podemos pensar que estamos manteniendo los colores viejos pero agregando una *bandera* (*flag*) por cada propiedad global de conectividad negativa. Nos referimos a la bandera asociada un determinado $C \in \mathcal{C}^-$ como la C -*flag*. La función de peso se mantiene esencialmente idéntica, simplemente ignorando las banderas. La función de verificación también, pero además hace el siguiente cambio: si el vértice v recibe un color a en su primera componente y este a está contenido en alguna $C \in \mathcal{C}^-$, la función de verificación ahora debe también asegurarse de que v no tenga ningún vecino con un color $a' \in C$ en su primera componente y con una diferente C -flag. Si un vértice recibe un color a en su primera componente y este a no está contenido en ninguna $C \in \mathcal{C}^-$, sus banderas no tienen importancia alguna. Por otro lado también agregamos propiedades globales de tamaño para garantizar que para cada $C \in \mathcal{C}^-$ exista al menos un vértice cuya primera componente del color esté en C y cuya C -flag sea 0, y además exista otro con esta característica pero tal que su C -flag sea 1. Notemos que la nueva cantidad de colores es $q2^{|\mathcal{C}^-|}$, que es menor o igual a $q2^{2^q}$.

Como $\text{snec}_{d,q}(T) \leq n^{qdw}$, donde w es la *mim-width* del árbol binario de descomposición dado (ver Observación 5.1.1), podemos decir que si tenemos propiedades globales de conectividad positiva y negativa dadas por conjuntos $\mathcal{C}^+$ y $\mathcal{C}^-$ respectivamente, entonces la complejidad es $O\left(n^{2q2^{|\mathcal{C}^-|} + wdq2^{|\mathcal{C}^-|}(6|\mathcal{C}^+| + \omega + 5)}\right)$.

Corolario 5.1.4 ([103, Teorema 1.2]). *Sean d y q enteros positivos. Para un problema 1-localmente verificable con q colores, funciones de verificación y peso d -estables, propiedades globales de tamaño, y propiedades globales de conectividad positiva y negativa dadas por $\mathcal{C}^+, \mathcal{C}^- \subseteq \mathcal{P}([1, q])$ respectivamente, existe un algoritmo que resuelve el problema en grafos con un árbol binario de descomposición dado en tiempo $n^{O(wdq(|\mathcal{C}^+|+1)2^{|\mathcal{C}^-|})}$, donde n es la cantidad de vértices del grafo de entrada y w es la *mim-width* del árbol binario de descomposición asociado.*

5.1.1. Extensión a distancias $r > 1$

En esta sección explicaremos cómo adaptar el algoritmo anterior a funciones d -estables a t distancias $r_1, \dots, r_t$, donde t está acotada por una constante (pero no hay restricciones sobre los valores de las distancias $r_1, \dots, r_t$). La idea clave aquí es usar el hecho de que la vecindad a una cierta distancia r de un vértice v en un grafo G , $N_G^r(v)$, equivale a la vecindad de v en la r -ésima potencia de G , $N_{G^r}(v)$ (ver Sección 1.5), pero

fundamentalmente que si la mim-width de un árbol binario de descomposición de G es w entonces la mim-width de ese mismo árbol para G^r es a lo sumo $2w$ (ver [121, Teorema 5]).

Sea t un entero positivo fijo. Sean $r_1, \dots, r_t$ enteros positivos. Para todo entero positivo r , definimos $\equiv_d^{v,r}$ y $\equiv_{d,q}^{v,r}$ como antes, sólo que ahora nos referimos al grafo G^r . Análogamente, definimos $\mathcal{R}_d^{v,r}$ y $\mathcal{R}_{d,q}^{v,r}$, así como también todo lo definido anteriormente para $\bar{v}$. También definimos la T_v^r -validez parcial de una tupla $(X, R^1, \dots, R^t)$ sólo considerando representantes $Y \subseteq \text{part}_q(T_{\bar{v}})$ para los cuales $Y \equiv_{d,q}^{\bar{v},r_i} R^i$ para todo $i \in [1, t]$. Para cada $v \in V(T)$, $R^i \in \mathcal{R}_d^{v,r_i}$, $R^i \in \mathcal{R}_{d,q}^{\bar{v},r_i}$, con $i \in [1, t]$, y $K' \in \text{part}_q(|T_v|)$, calculamos $M_v[R^1, R^1, \dots, R^t, R^t, K'] \subseteq \text{part}_q(T_v)$ con las mismas condiciones de antes y adicionalmente para cada X en este conjunto pedimos que $X \equiv_{d,q}^{v,r_i} R^i$ para cada $i \in [1, t]$. El requerimiento de tamaño para M_v se mantiene igual. El resto del algoritmo es esencialmente el mismo, sin cambios fundamentales. Observemos en particular que la Afirmación 5.1.3 puede usarse sin ningún cambio, ya que sólo tendríamos más restricciones sobre Y . Al computar $\mathcal{M}$, tenemos que considerar más posibilidades de combinaciones, pero luego del paso de reducción, el tamaño será el mismo que en la versión original, sólo cambia la complejidad temporal.

Explicaremos ahora la nueva complejidad temporal obtenida. Primero debemos calcular los grafos $G^{r_1}, \dots, G^{r_t}$ y los representantes correspondientes. Cada una de las matrices M_v tiene $\text{snec}_{d,q}(T)^{2t} n^q$ entradas. La cantidad de elementos en cada una de ellas sigue siendo a lo sumo $2^{|C|} \cdot \text{snec}_{d,q}(T)^{2|C|}$. El caso base también sigue siendo de tiempo constante (porque estamos suponiendo que t está acotada por una constante). Cuando v no es una hoja, calculamos $\mathcal{M}$ y lo reducimos con la Afirmación 5.1.3 en tiempo $O(n^q \cdot \text{snec}_{d,q}(T)^{4t} \cdot (2^{|C|} \cdot \text{snec}_{d,q}(T)^{2|C|})^2 \cdot \text{snec}_{d,q}(T)^{2|C|} \cdot (n^2 + \text{snec}_{d,q}(T)^{\omega-1}))$. Finalmente, la complejidad total resulta $O(n^{2q} \cdot \text{snec}_{d,q}(T)^{6|C|+6t} \cdot (n^2 + \text{snec}_{d,q}(T)^{\omega-1}))$. Usando la desigualdad sobre $\text{snec}_{d,q}(T)$, obtenemos $n^{O(wdq(|C|+t))}$. Observemos que podemos incluir propiedades globales de conectividad negativa de la misma forma que en el Corolario 5.1.4. Entonces obtenemos el siguiente resultado:

Teorema 5.1.5 ([103, Corolario 5.4]). *Sean d, q y t enteros positivos. Para un problema localmente verificable con q colores, funciones de verificación y de peso que son d -estables a t distancias, propiedades globales de tamaño, y propiedades globales de conectividad positiva y negativa dadas por $\mathcal{C}^+, \mathcal{C}^- \subseteq \mathcal{P}([1, q])$ respectivamente, existe un algoritmo que resuelve el problema en grafos con un árbol binario de descomposición dado en tiempo $n^{O(wdq(|C^+|+t)2^{|C^-|})}$, donde n es la cantidad de vértices del grafo de entrada y w es la mim-width del árbol binario de descomposición asociado.*

Es importante destacar que la cantidad t de distancias afecta a la complejidad, pero no los valores concretos de cada distancia r_i . Esto nos permite considerar problemas en los cuales, por ejemplo, cada vértice de color rojo está a distancia a lo sumo $\frac{n}{2}$ de un vértice de color azul.

5.1.2. Extensión a propiedades globales de composición de condiciones locales

En el algoritmo presentado mostramos cómo incluir propiedades globales de tamaño, pero notemos que con una ligera modificación este algoritmo se puede adaptar a

propiedades globales de composición de condiciones locales cuya función asociada sea d -estable y cuyo monoide sea de tamaño polinomial.

Sean $(M, \otimes)$ un monoide conmutativo, X un subconjunto de M y $f: V(G) \times \text{COLORS} \times \mathbb{N}^q \rightarrow M$ una función d -estable. Vamos a suponer además que $|M|$ es polinomial en $|V(G)|$. Sea $m \in M$. Consideremos el problema localmente verificable donde el coloreo propio c que buscamos además debe cumplir que $m = \bigotimes_{w \in V(G)} f(w, c(w), (|N(w) \cap c^{-1}(a_1)|, \dots, |N(w) \cap c^{-1}(a_q)|))$.

Vamos a realizar las siguientes modificaciones sobre el algoritmo original. Toda aparición de “ $|X|$ ”, con $X \in \text{part}_q(T_v)$ para algún nodo v de T , se reemplaza por “ $\bigotimes_{w \in T_v} f(w, j_w, (|N(w) \cap (X_1 \cup Y_1)|, \dots, |N(w) \cap (X_q \cup Y_q)|))$ ”, donde j_w se define como el entero en $[1, q]$ tal que $w \in X_{j_w}$. Reemplazamos análogamente para $Y \in \text{part}_q(T_{\bar{v}})$. Toda aparición de K se reemplaza por m , y toda aparición de algún otro $K' \in \text{part}_q(|T_v|)$ se reemplaza por m' con $m' \in M$. La sumas de tamaños deben ser cambiadas por el operador del monoide $\otimes$.

Ahora el tamaño de $\mathcal{M}$ es a lo sumo $|M|^2 \cdot \text{sne}_{d,q}(T)^4 \cdot (2^{|\mathcal{C}|} \text{sne}_{d,q}(T)^{2^{|\mathcal{C}|}})^2$ porque existen a lo sumo $|M|^2 \cdot \text{sne}_{d,q}(T)^4$ pares de tuplas (R_a, R'_a, m^a) y (R_b, R'_b, m^b) que son (R, R', m') -compatibles. Además, la matriz M_v ahora tiene a lo sumo $\text{sne}_{d,q}(T)^2 \cdot |M|$ entradas. Cada entrada de M_v puede calcularse en tiempo $O(|M|^2 \cdot \text{sne}_{d,q}(T)^{6^{|\mathcal{C}|+4}} \cdot (n^2 + \text{sne}_{d,q}(T)^{\omega-1}))$. Así, obtenemos que la complejidad total es $O(n \cdot |M|^3 \cdot \text{sne}_{d,q}(T)^{6^{|\mathcal{C}|+6}} \cdot (n^2 + \text{sne}_{d,q}(T)^{\omega-1}))$.

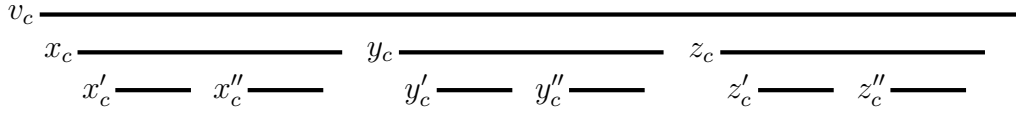
Agregando las modificaciones para propiedades de conectividad negativa y mayores distancias, obtenemos una complejidad de $n^{O(wdq(|\mathcal{C}^+|+t)2^{|\mathcal{C}^-|})}$, porque estamos suponiendo que $|M|$ es polinomial en n (es decir, que $|M| \leq n^k$ para algún entero positivo fijo k). Entonces la complejidad no cambia significativamente al implementar esta modificación.

Teorema 5.1.6. *Sean d, q y t enteros positivos. Para un problema localmente verificable con q colores, funciones de verificación y de peso que son d -estables a t distancias, una propiedad global de composición de condiciones locales cuya función asociada es d -estable y cuyo monoide es de tamaño polinomial, y propiedades globales de conectividad positiva y negativa dadas por $\mathcal{C}^+, \mathcal{C}^- \subseteq \mathcal{P}([1, q])$ respectivamente, existe un algoritmo que resuelve el problema en grafos con un árbol binario de descomposición dado en tiempo $n^{O(wdq(|\mathcal{C}^+|+t)2^{|\mathcal{C}^-|})}$, donde n es la cantidad de vértices del grafo de entrada y w es la mim-width del árbol binario de descomposición asociado.*

5.2. Límites

Como el problema MÍNIMO CONJUNTO DOMINANTE es $W[1]$ -difícil parametrizado por mim-width [87], es poco probable que exista un algoritmo FPT para resolver todos los problemas 1-localmente verificables con funciones 1-estables y cantidad constante de colores. Más aún, mostraremos que es poco probable que los resultados de la Sección 5.1 puedan extenderse a problemas 1-localmente verificables con menos restricciones. En particular, analizaremos las siguientes tres situaciones: cantidad de colores no acotada por una constante, funciones de peso contadoras de colores no estables, y funciones de verificación contadoras de colores no estables (ver Tabla 5.1).

Como probamos en la Sección 4.4, si pudiéramos resolver cualquier problema 1-localmente verificable con cantidad no constante de colores y funciones 1-estables en

Figura 5.1: Gadget de cláusula para $c = (x, y, z)$.

grafos completos (que tienen mim-width igual a 1), entonces probaríamos que $P=NP$.

Consideremos el problema MAX-CUT, el cual es NP-difícil en grafos de intervalo [8] pero, como vimos en la Sección 4.4, podemos modelarlo como un problema 1-localmente verificable con dos colores, una función de verificación 1-estable y una función de peso contadora de colores no estable. Por lo tanto, no es esperable encontrar un algoritmo eficiente parametrizado por mim-width para resolver problemas 1-localmente verificables con cantidad constante de colores, función de verificación 1-estable pero función de peso no estable.

Para concluir, vamos a mostrar que no podemos resolver eficientemente todos los problemas 1-localmente verificables con cantidad constante de colores, función de peso 1-estable y función de verificación contadora de colores no estable, salvo que $P=NP$. Para esto definiremos un nuevo problema con estas características y probaremos que es NP-difícil en grafos de intervalo (los cuales tienen mim-width a lo sumo 1).

Sea $COLORS = \{1, \dots, 7, T, t, F, f\}$. Definimos los pesos como $(WEIGHTS, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$ y $w'(v, a, \mu) = 0$. Notemos que $\deg(v)$ (el grado de un vértice v) es igual a la suma de la cantidad de vecinos de todos los colores. Para un vértice v , un color $a \in COLORS$ y un vector μ de enteros no negativos indexado por $COLORS$, definimos:

$$\begin{aligned}
 check'(v, a, \mu) = & \\
 & (a = 1 \Leftrightarrow \deg(v) = 2) \\
 & \wedge (a = 2 \Leftrightarrow \deg(v) > 2 \wedge \mu_1 \geq 1) \\
 & \wedge (a = 3 \Leftrightarrow \deg(v) > 2 \wedge \mu_1 = 0 \wedge \mu_2 \geq 2) \\
 & \wedge (a = 4 \Leftrightarrow \deg(v) > 2 \wedge \mu_1 = 0 \wedge \mu_2 = 1) \\
 & \wedge (a = 5 \Leftrightarrow \deg(v) > 2 \wedge \mu_1 = 0 \wedge \mu_2 = 0 \wedge \mu_3 = \mu_4) \\
 & \wedge (a = 6 \Leftrightarrow \deg(v) > 2 \wedge \mu_1 = 0 \wedge \mu_2 = 0 \wedge \mu_3 \neq \mu_4 \wedge \mu_5 > 1) \\
 & \wedge (a = 7 \Leftrightarrow \deg(v) > 2 \wedge \mu_1 = 0 \wedge \mu_2 = 0 \wedge \mu_3 \neq \mu_4 \wedge \mu_5 = 1 \wedge \sum_{i \in \{7, T, t, F, f\}} \mu_i = 2) \\
 & \wedge (a = 5 \Rightarrow \mu_T = 1) \\
 & \wedge (a = 6 \Rightarrow \mu_T = \mu_t \wedge \mu_F = \mu_f) \\
 & \wedge (a = 7 \Rightarrow \mu_t = \mu_T = 1 \vee \mu_f = \mu_F = 1).
 \end{aligned}$$

Afirmamos que este problema es NP-difícil incluso cuando nos restringimos a grafos de intervalo. Para esto vamos a reducir polinomialmente un problema llamado POSITIVE 1-IN-3 SAT. Este problema, que es NP-completo [168], es una variante de 3-SAT en la cual todos los literales son positivos y una asignación de verdad es satisfactoria si asigna exactamente un literal verdadero en cada cláusula.

Dada una instancia I de POSITIVE 1-IN-3-SAT con un conjunto de cláusulas C (con $C \neq \emptyset$) y un conjunto de variables X , construimos el grafo de intervalo G_I de la siguiente manera. Para cada cláusula $c = (x, y, z)$ creamos una copia del *gadget de cláusula* ilustrado en la Figura 5.1. Llamamos *vértice cláusula* a v_c , *vértice variable* a cada x_c, y_c y z_c , y *vértice valor* a cada $x'_c, x''_c, y'_c, y''_c, z'_c$ y z''_c . Todos los gadgets de

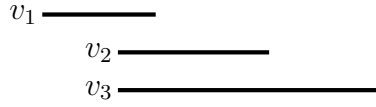


Figura 5.2: Left gadget.

Figura 5.3: Ejemplo de una representación de los intervalos de G_I , donde I consiste de tres cláusulas $c_1 = (x_1, x_2, x_3)$, $c_2 = (x_4, x_5, x_6)$ y $c_3 = (x_7, x_6, x_8)$.

cláusula son dispuestos uno al lado del otro, en cualquier orden, de modo que no se superpongan. Para cada variable $x \in X$, hacemos lo siguiente. Consideremos todo par de cláusulas c y c' tales que ambas contienen a x y cuyos gadgets están dispuestos de modo que no hay ningún otro gadget entre ellos que se corresponda con otra cláusula que contiene a x . Sin pérdida de generalidad, supongamos que el gadget de c está a la izquierda del de c' . Agregamos un intervalo cuyo extremo izquierdo esté en el intervalo correspondiente a x'_c y cuyo extremo derecho esté en el intervalo correspondiente a $x'_{c'}$. Los vértices correspondientes a estos intervalos se llamarán *vértices conectores*. Además, agregamos el *gadget izquierdo* ilustrado en la Figura 5.2 a la izquierda de todos los intervalos dispuestos hasta el momento, sin superponerse con ninguno de ellos. Para cada cláusula $c \in C$ agregamos otros dos intervalos de la siguiente manera. Agregamos un intervalo cuyo extremo izquierdo esté en el intervalo de v_2 pero no en el intervalo de v_1 , y cuyo extremo derecho esté en el intervalo correspondiente a v_c pero a la izquierda de cualquier intervalo correspondiente a un vértice variable en el gadget de la cláusula c . El vértice correspondiente a este intervalo se llamará *vértice de apertura de c* . Agregamos otro intervalo cuyo extremo izquierdo esté en el intervalo correspondiente a v_3 pero no en el intervalo correspondiente a v_2 , y cuyo extremo derecho esté en el intervalo correspondiente a v_c pero a la derecha de cualquier intervalo correspondiente a un vértice variable en el gadget de la cláusula c . El vértice correspondiente a este intervalo se llamará *vértice de cierre de c* . En la Figura 5.3 podemos ver un ejemplo con tres cláusulas, en el cual una sola variable aparece dos veces (x_6).

Notemos que G_I puede ser construido en tiempo polinomial a partir de una instancia I . Afirmamos que G_I admite un coloreo propio si y sólo si I es satisfacible.

Primero, supongamos que tenemos un coloreo propio col de G_I . Es fácil ver que v_1 es el único vértice de color 1 (porque es el único vértice de grado 2) y los vértices v_2 y v_3 son los únicos vértices de color 2 (porque son los únicos vecinos de un vértice de color 1). Luego, todos los vértices de apertura tienen color 3 y todos los de cierre tienen color 4, y ningún otro vértice puede tener estos colores. Ahora notemos que cada vértice cláusula es adyacente a la misma cantidad de vértices de apertura que de cierre, pero esta propiedad no vale para ningún vértice variable, valor ni conector (en particular, un vértice conector $u_{x,c,c'}$ es adyacente a la misma cantidad de vértices de apertura y cierre que el vértice valor x''_c , en cuyo intervalo está el extremo izquierdo del intervalo de $u_{x,c,c'}$). Entonces, cada vértice cláusula tiene color 5 y ningún otro vértice lo tiene. Como cada vértice conector tiene al menos dos vértices cláusula en su vecindad, los vértices conectores reciben color 6. Todos los vértices variable y valor tienen exactamente un vecino de color 5, por lo tanto reciben color 7, T, t, F o f.

Restricciones sobre <i>check</i> y <i>w</i>	Restricciones sobre $ \text{COLORS} $	
	Constante	Polinomial
<i>d</i> -estable	XP ⁽¹⁾ , W[1]-h	para-NP-h
Contadora de colores	para-NP-h	para-NP-h

Tabla 5.1: Resumen de los resultados principales sobre la complejidad (parametrizada por la mim-width de un árbol binario de descomposición dado) de problemas 1-localmente verificables sin etiquetas en las aristas, bajo distintas restricciones sobre la cantidad de colores y las funciones de verificación y peso. (1) dado un árbol binario de descomposición adecuado. Aquí, cuando indicamos para-NP-h (o W[1]-h), estamos diciendo que existe un problema con esas condiciones que es para-NP-h (o W[1]-h) parametrizado por mim-width.

Luego, tenemos que cada vértice valor tiene exactamente un vecino cuyo color no está en $\{1, \dots, 6\}$, mientras que los vértices variable tienen dos de ellos. Por lo tanto los vértices variable reciben color 7, y los vértices valor reciben T, t, F o f. Para cada vértice variable, el conjunto de colores asignados a sus dos vecinos que son vértices valor es o bien $\{T, t\}$ o bien $\{F, f\}$. Consideremos ahora los vértices valor en la vecindad de algún vértice conector w fijo. Ya vimos que todos ellos, excepto por aquellos cuyos intervalos contienen los extremos del intervalo de w , aparecen en pares coloreados o bien con T y t, o bien con F y f. Como la cantidad de vértices valor en $N(w)$ que reciben color T tiene que ser la misma que la cantidad de vértices valor en $N(w)$ de color t (y análogamente para F y f), sigue que los dos vértices valor cuyos intervalos contienen a los extremos del intervalo de w tienen que tener los colores T y t o los colores F y f. Esto implica que para cada variable $x \in X$, o bien para cada cláusula c que contiene a x tenemos $col(x'_c), col(x''_c) \in \{T, t\}$, o bien para cada cláusula c que contiene a x tenemos $col(x'_c), col(x''_c) \in \{F, f\}$. Finalmente, cada vértice de color 5 tiene exactamente un vecino de color T. Esto significa que en la vecindad de cada vértice cláusula hay exactamente un vértice variable cuyos vértices valor adyacentes tienen colores $\{T, t\}$, y los restantes dos vértices variables tienen sus vértices valor adyacentes coloreados con $\{F, f\}$. Asignamos el valor TRUE a una variable $x \in X$ si todos los vértices valor adyacentes a los vértices variable correspondientes al literal de x están coloreados con T o t, y asignamos el valor FALSE si no. Afirmamos que esta asignación es satisfactoria para I . En efecto, cada variable recibe un único valor (o bien TRUE o bien FALSE, ya que el conjunto de colores correspondiente a los vértices valor es o bien $\{T, t\}$ o bien $\{F, f\}$), y cada cláusula tiene exactamente una variable asignada a TRUE.

Supongamos ahora que tenemos una instancia I de POSITIVE 1-IN-3 SAT que es satisfacible. Sea f una función del conjunto de variables a $\{\text{TRUE}, \text{FALSE}\}$ que asigna valores a las variables de modo de satisfacer la instancia. Vamos a construir un coloreo propio del grafo G_I . Asignamos el color 1 a v_1 , el color 2 a v_2 y v_3 , el color 3 a cada vértice de apertura, el color 4 a cada vértice de cierre, el color 5 a cada vértice cláusula, el color 6 a cada vértice conector, y el color 7 a cada vértice variable. Para cada vértice valor x'_c , si $f(x) = \text{TRUE}$ entonces asignamos a x'_c el color T, de lo contrario asignamos a x'_c el color F. Análogamente, para cada vértice valor x''_c , si $f(x) = \text{TRUE}$ asignamos a x''_c color t, de lo contrario asignamos a x''_c el color f. Es sencillo verificar que este es un coloreo propio.

Capítulo 6

Esquemas de aproximación de tiempo polinomial (PTAS)

Un *esquema de aproximación de tiempo polinomial* (en inglés, *Polynomial-Time Approximation Scheme*, abreviado *PTAS*) para un problema de optimización Π es un algoritmo que, dada una instancia I de Π y un número real $\epsilon > 0$, devuelve una solución aproximada de Π para la instancia I , cuyo valor es a lo sumo $1 + \epsilon$ veces el valor de la solución óptima (en el caso de que Π sea un problema de minimización), o al menos $1 - \epsilon$ veces el valor de la solución óptima (en el caso de que Π sea de maximización). Adicionalmente, la complejidad temporal de este algoritmo debe ser polinomial en el tamaño de I para cada valor fijo de ϵ .

Por ejemplo, en [17] se presenta un PTAS para el problema de MÁXIMO CONJUNTO INDEPENDIENTE en grafos planares que, para cada entero positivo k , produce una solución de tamaño al menos $\frac{k}{k+1}$ veces el valor óptimo y cuya complejidad temporal es $O(k8^kn)$, siendo n la cantidad de vértices del grafo planar de entrada. En otras palabras, para cada número real $\epsilon > 0$, se puede encontrar en tiempo $O\left(\frac{1}{\epsilon}8^{\frac{1}{\epsilon}}n\right)$ una solución a este problema que sea al menos $1 - \epsilon$ veces la óptima. Por otro lado, existen problemas para los cuales no podemos decir esto: el problema de colorear un grafo con la menor cantidad posible de colores no admite un PTAS en grafos planares, salvo que $P=NP$. Esto se debe a que sabemos que dichos grafos son 4-coloreables pero determinar si son 3-coloreables es un problema NP-completo [71], y por lo tanto no podemos aspirar a obtener aproximaciones con un factor mejor que $\frac{4}{3}$ en tiempo polinomial.

La *técnica de Baker*, descrita por primera vez por Brenda Baker [17], es un método para obtener esquemas de aproximación de tiempo polinomial en grafos planares. La idea general consiste en descomponer el grafo en varias *capas*, de modo tal que cada una de ellas tenga treewidth acotada por una constante determinada y por lo tanto sea “fácil” resolver un determinado problema en esos subgrafos. Luego, las soluciones parciales de cada capa se combinan para así obtener una aproximación de la solución óptima en el grafo original. Este método resulta de gran utilidad ya que muchos problemas relevantes son NP-difíciles en grafos planares (por ejemplo, MÍNIMO CONJUNTO DOMINANTE y MÁXIMO CONJUNTO INDEPENDIENTE [92]), y ha sido ampliamente utilizado para obtener PTAS en grafos planares de diversos problemas [17, 138, 169].

En este capítulo analizaremos bajo qué condiciones podemos aplicar la técnica de Baker a problemas 1-localmente verificables, obteniendo así PTAS en grafos planares para todos aquellos problemas que satisfagan estas condiciones.

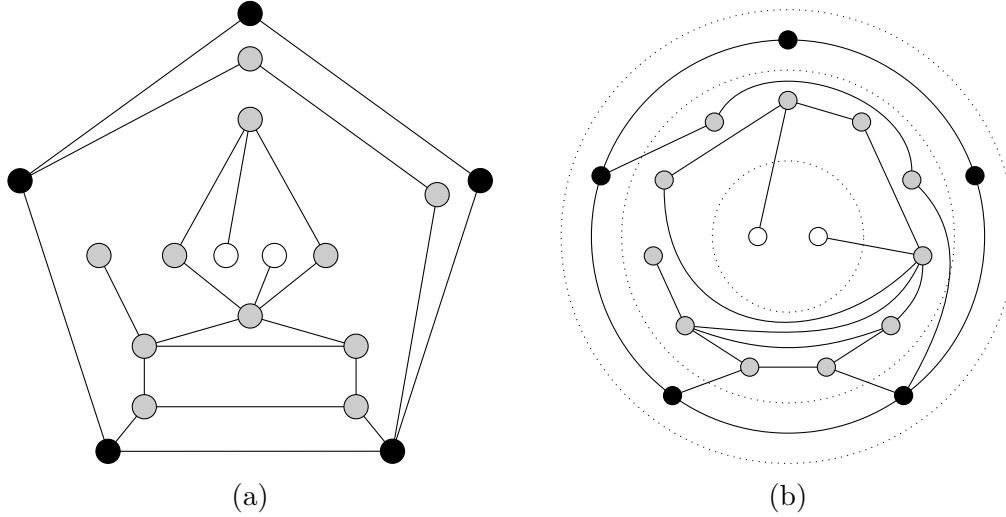


Figura 6.1: (a) Grafo planar y su descomposición en niveles, donde el nivel 1 es representado en color negro, el nivel 2 en gris y el nivel 3 en blanco. (b) Otra representación del mismo grafo donde los niveles están dispuestos circularmente. Todos los grafos planares admiten una representación de esta forma, y la misma facilita la comprensión de la técnica de Baker.

6.1. Preliminares

Consideremos un grafo planar G y alguna representación plana del mismo. El *nivel 1* de G se define como el conjunto V_1 de vértices de G que se encuentran en la cara externa de la representación plana de G dada. A partir de entonces, el *nivel x* de G , con $x > 1$, se define como el conjunto $V_x \subseteq V(G) - \bigcup_{i=1}^{x-1} V_i$ de los vértices que se encuentran en la cara externa de la representación plana de G sin los vértices de los niveles anteriores. La Figura 6.1 ejemplifica la descomposición en niveles de un grafo planar. Nótese que no puede haber aristas entre vértices de niveles que no sean consecutivos. En tiempo lineal podemos determinar si un grafo es planar, dar una representación plana y encontrar sus niveles [58, 114, 136].

Un grafo se llama *k -exterior-planar* (*k -outerplanar* en inglés) si admite una representación plana con a lo sumo k niveles. Estos grafos tienen la propiedad de que su treewidth es a lo sumo $3k - 1$ [34, Teorema 83]. Por lo tanto, muchos problemas son resolubles en tiempo polinomial en ellos (para cada valor fijo de k), en particular los problemas 1-localmente verificables con cantidad polinomial de colores y que admiten un sistema de vecindades parciales polinomial (como vimos en el Capítulo 3).

A continuación definiremos la descomposición en *capas* de G , siguiendo la notación utilizada en [138]. Sea m la cantidad de niveles de G , y sean k y s enteros tales que $2 \leq k < m$ y $2 \leq s \leq k + 1$. Notemos $r_s = \lceil \frac{m-s}{k} \rceil$. Para todo $i \in \llbracket 0, r_s \rrbracket$, la capa $W(i, s)$ se define como:

$$W(i, s) = \begin{cases} \bigcup_{j=1}^s V_j & \text{si } i = 0 \\ \bigcup_{j=-1}^k V_{(i-1)k+s+j} & \text{si } 1 \leq i < r_s \\ \bigcup_{j=(r_s-1)k+s-1}^m V_j & \text{si } i = r_s. \end{cases}$$

De esta manera, la capa con $i = 0$ tiene s niveles, la capa con $i = r_s$ tiene al menos 3 y a lo sumo $k + 2$ niveles, y las demás tienen exactamente $k + 2$ niveles cada una. Por

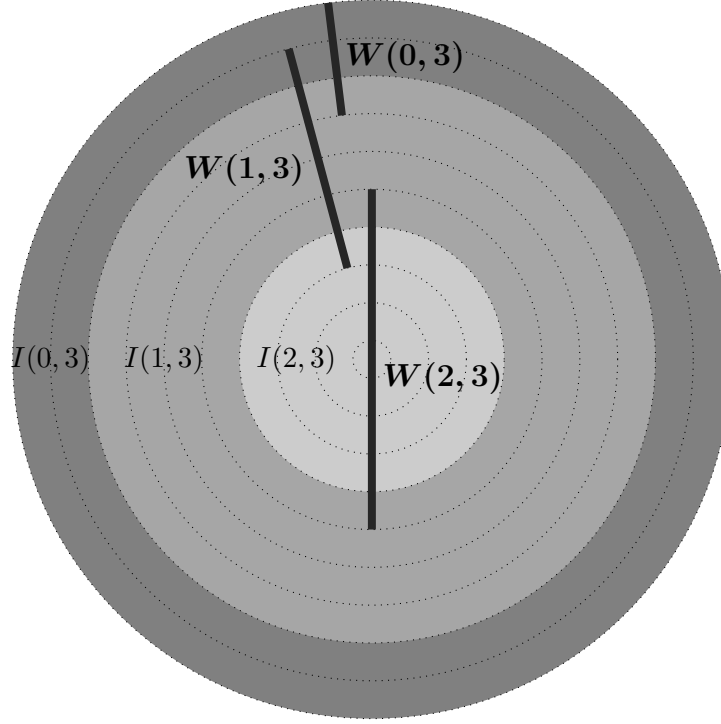


Figura 6.2: Representación de la superposición de las capas, en un grafo con 10 niveles (se omiten los vértices y las aristas en el dibujo), $k = 4$ y $s = 3$. La capa $W(0, 3)$ incluye los niveles de 1 a 3, la capa $W(1, 3)$ los niveles de 2 a 7, y $W(2, 3)$ de 6 a 10, como se indica con los segmentos en línea gruesa. La frontera $B(0, 3)$ es el nivel 3, $B(1, 3)$ consiste en los niveles 2 y 7, y $B(2, 3)$ es el nivel 6. Los conjuntos de vértices internos $I(0, 3)$, $I(1, 3)$ y $I(2, 3)$ están sombreados en distintas tonalidades de gris.

lo tanto, como $s < k + 2$, cada uno de los grafos inducidos por una capa es $(k + 2)$ -outerplanar. Notemos, además, que las capas consecutivas se solapan en dos niveles. El conjunto de vértices en la *frontera* de la capa $W(i, s)$, notado $B(i, s)$, se define como:

$$B(i, s) = \begin{cases} V_s & \text{si } i = 0 \\ V_{(i-1)k+s-1} \cup V_{ik+s} & \text{si } 1 \leq i < r_s \\ V_{(r_s-1)k+s-1} & \text{si } i = r_s. \end{cases}$$

También se define el conjunto $I(i, s)$ de vértices *internos* de $W(i, s)$ como aquellos que no están en la frontera, es decir, $I(i, s) = W(i, s) - B(i, s)$. Es fácil ver que, para todo $s \in \llbracket 2, k + 1 \rrbracket$ y todo $i \in \llbracket 0, r_s \rrbracket$, los vecinos de los vértices internos de una capa deben pertenecer a dicha capa. Formalmente, $N[I(i, s)] \cap (V(G) - W(i, s)) = \emptyset$. La Figura 6.2 muestra un ejemplo gráfico de todas estas definiciones.

Para problemas de minimización (resp. maximización), fijado un entero k , la técnica de Baker consiste esencialmente en los siguientes pasos:

1. Obtener una representación plana de G y sus niveles, digamos $V_1, \dots, V_m$. Vamos a suponer que $m > k$, ya que de lo contrario el grafo sería k -outerplanar y podríamos dar directamente una solución exacta al problema.
2. Para cada $s \in \llbracket 2, k + 1 \rrbracket$ y cada $i \in \llbracket 0, r_s \rrbracket$, obtener la capa $W(i, s)$ y calcular la

solución óptima en $G[W(i, s)]$ pero no del problema original sino de un *problema asociado*. A esta solución la llamaremos $S(i, s)$.

3. Para cada $s \in \llbracket 2, k+1 \rrbracket$, *combinar* todas las soluciones $S(i, s)$, con $i \in \llbracket 0, r_s \rrbracket$, obteniendo S_s .
4. Devolver la solución de menor (resp. mayor) peso entre todas las soluciones $S_2, \dots, S_{k+1}$.

Hay que elegir adecuadamente el problema asociado y el algoritmo de combinación, de modo que todas las $S_2, \dots, S_{k+1}$ sean efectivamente soluciones válidas del problema y además se puedan calcular en tiempo polinomial. Como veremos en las próximas secciones, en nuestro caso concreto vamos a asegurarnos de que $S(i, s)$ sea tal que se cumpla la propiedad local para cada vértice interno de la capa, y que la combinación de dos soluciones cambie sólo los colores de los niveles solapados y este cambio no invalide la propiedad local en ningún vértice.

Llamemos n a la cantidad de vértices de G y $n_{i,s}$ a la cantidad de vértices en cada capa $W(i, s)$. Sea $T(n_{i,s}, k)$ una cota superior del tiempo para resolver el problema asociado en $G[W(i, s)]$. Entonces podemos decir que la complejidad total de este procedimiento es $O\left(nk + \sum_{s=2}^{k+1} \sum_{i=0}^{r_s} T(n_{i,s}, k)\right)$. En particular, si existen funciones f, g tales que $g(k) \geq 1$ y $T(n_{i,s}, k) = f(k) \cdot n_{i,s}^{g(k)}$ para todos los posibles pares (i, s) , entonces la complejidad total resulta $O(k \cdot f(k) \cdot (2n)^{g(k)})$, porque para cada $s \in \llbracket 2, k+1 \rrbracket$ tenemos $\sum_{i=0}^{r_s} n_{i,s} \leq 2n$ y porque $x^z + y^z \leq (x+y)^z$ para todos x, y, z mayores o iguales a 1.

6.2. Restricciones consideradas

En primer lugar, como los problemas van a ser resueltos en grafos de treewidth acotada, construiremos sobre las restricciones consideradas en el Capítulo 3. Es decir, vamos a considerar problemas 1-localmente verificables con cantidad polinomial de colores y que admiten un sistema de vecindades parciales polinomial, y que además satisfacen las restricciones que definiremos a continuación. Consideraremos problemas 1-localmente verificables sin propiedades globales.

En particular, como no existe un PTAS para el problema de colorear un grafo con la menor cantidad posible de colores en grafos planares (salvo que $P=NP$), pero sí existe uno para MÍNIMO CONJUNTO DOMINANTE [138], las restricciones consideradas tienen que poder distinguir lo que intuitivamente reconocemos como problemas de dominación de los problemas clásicos de coloreo.

Dado que nuestro objetivo es obtener algoritmos aproximados para un problema, éste debe admitir un conjunto de pesos contenido en los números reales, cuyo operador $\oplus$ sea la suma usual. Por el mismo motivo, también es necesario hacer una distinción entre problemas de *minimización* y problemas de *maximización*. Vamos a considerar también que la función de peso sólo depende del vértice y del color que ese vértice recibe (es decir, no tiene en cuenta el coloreo de su vecindad), y por ello vamos a escribir simplemente $w_{v,a}$ para referirnos al peso de un vértice v con color a . A este tipo de función de peso la llamaremos *simple*.

Para poder aplicar la técnica de Baker a un problema localmente verificable de minimización, necesitamos poder *combinar* coloreos propios de subgrafos inducidos de

G . Una situación bajo la cual es posible hacer esto es cuando, para cada $v \in V(G)$, para todo par de colores $a, a' \in L_v$ existe un color $comb(v, a, a') \in L_v$ tal que, en cualquier coloreo propio de G donde el color de v es a o a' , podemos reemplazar el color de v por $comb(v, a, a')$ y obtener nuevamente un coloreo propio de G . De alguna manera esto impone una cierta jerarquía entre los colores, la cual se da naturalmente en, por ejemplo, problemas de dominación pero no se da en el problema de k -coloreo. Asimismo, hay que considerar alguna condición sobre el costo de asignar el color $comb(v, a, a')$ a v . En este caso, vamos a considerar la restricción $W_{v, comb(v, a, a')} \leq W_{v, a} + W_{v, a'}$. Cuando todo esto ocurra, diremos que el problema cuenta con una *jerarquía de colores*.

Respecto a los problemas localmente verificables de maximización, vamos a pedir que exista un subconjunto de *colores neutrales* $A \subseteq \text{COLORS}$ tal que:

1. Para todo vértice v y todo coloreo c de $N_G[v]$ tal que $c(v) \in A$, si $check(v, c) = \text{TRUE}$ entonces $check(v, c') = \text{TRUE}$ para cualquier otro coloreo c' con $c'(v) = c(v)$, $c'(u) = c(u)$ para todo $u \in N_G(v)$ tal que $c(u) \notin A$ y $c'(u) \in A \cap L_u$ para todo $u \in N_G(v)$ tal que $c(u) \in A$. En otras palabras, cuando v tiene un color neutral, podemos reemplazar los colores de sus vecinos con colores neutrales por cualquier otra combinación válida de colores neutrales.
2. Para todo coloreo propio c de G y para todo $W \subseteq V(G)$, existe al menos un coloreo propio c' de G tal que $c'(w) = c(w)$ para todo $w \in W$ y $c'(v) \in A$ para todo $v \notin W$. Es decir, existe al menos una forma de extender el coloreo parcial $c|_W$ a todo el grafo, utilizando sólo colores neutrales y obteniendo también un coloreo propio.
3. Existe $\alpha \geq 0$ tal que $W_{v, a} = \alpha$ para todo $v \in V(G)$ y $a \in A \cap L_v$.

Esta condición puede resultar un tanto rebuscada a primera vista, pero se cumple para varios problemas de maximización. Por ejemplo, en MÁXIMO CONJUNTO INDEPENDIENTE, aquel color que representa a los vértices fuera del conjunto independiente es un color neutral, puesto que siempre podemos extender un conjunto independiente a todo el grafo al no agregar vértices al conjunto independiente, es decir, se cumple la condición 2, y las condiciones 1 y 3 se satisfacen trivialmente. Veremos más ejemplos en el Capítulo 7.

6.3. Problemas de minimización

Sea k un entero mayor o igual a 2. Consideremos un problema 1-localmente verificable Π con $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{R}^{+\infty}, \leq, +)$ y que además satisface las restricciones arriba mencionadas para problemas de minimización.

Sea G el grafo planar de entrada y sea m la cantidad de niveles de alguna representación plana de G . Si $m \leq k$ entonces G es k -outerplanar y podemos aplicar directamente el algoritmo del Capítulo 3 para dar una respuesta exacta al problema en tiempo polinomial. Por lo tanto, a partir de ahora supondremos que $m > k$.

Para todo $s \in \llbracket 2, k+1 \rrbracket$, notemos $r_s = \lceil \frac{m-s}{k} \rceil$. Para todo $s \in \llbracket 2, k+1 \rrbracket$ y todo $i \in \llbracket 0, r_s \rrbracket$, sea $S(i, s)$ una solución óptima del *problema asociado* Π' , el cual tiene como

entrada al grafo $G[W(i, s)]$, su función de verificación $check'$ es tal que:

$$check'(v, c) = \begin{cases} check(v, c) & \text{si } v \in I(i, s), \\ \text{TRUE} & \text{si no,} \end{cases}$$

siendo $check$ la función de verificación de Π , y en todo lo demás Π' es análogo a Π . En otras palabras, $S(i, s)$ es un coloreo de costo mínimo que cumple la función de verificación de Π para los vértices internos de la capa, pero no necesariamente para los de la frontera. Notemos que si existe un coloreo propio c de G para el problema Π , entonces existen los coloreos $S(i, s)$, ya que $c|_{W(i, s)}$ es un coloreo propio de $G[W(i, s)]$ para el problema Π' . Como estamos suponiendo que Π tiene cantidad polinomial de colores y admite un sistema de vecindades parciales polinomial, entonces Π' también tiene estas características y por lo tanto es XP parametrizado por treewidth. Además, como $G[W(i, s)]$ es k -outerplanar, podemos determinar $S(i, s)$ en tiempo polinomial.

Dados $A, B \subseteq V(G)$ y dos coloreos $c_1: A \rightarrow \text{COLORS}$ y $c_2: B \rightarrow \text{COLORS}$, definimos la *combinación* de c_1 y c_2 como el coloreo $c: A \cup B \rightarrow \text{COLORS}$ tal que:

$$c(v) = \begin{cases} c_1(v) & \text{si } v \in A - B, \\ c_2(v) & \text{si } v \in B - A, \\ comb(v, c_1(v), c_2(v)) & \text{si } v \in A \cap B. \end{cases}$$

Para todo $s \in \llbracket 2, k+1 \rrbracket$, sea S_s la combinación de todos los coloreos $S(i, s)$, con $i \in \llbracket 0, r_s \rrbracket$. Esto es, cada vértice v que recibe un color a en un coloreo parcial y un color a' en otro coloreo parcial, en S_s recibe el color $comb(v, a, a')$, y todos los vértices que reciban un único color lo mantienen. Notemos que este coloreo está bien definido y además es un coloreo propio en G para el problema Π . En efecto, esto se debe a que la función de verificación $check$ se satisface en cada vértice interno de cada capa, a que cada vértice es interno en exactamente una capa y a que (por las restricciones del problema) cambiar el color a de un vértice v por $comb(v, a, a')$ sigue produciendo un coloreo propio. Además, este coloreo se puede obtener en tiempo polinomial.

Sea D un coloreo propio óptimo en G . Para todo $s \in \llbracket 2, k+1 \rrbracket$ tenemos que (justificación debajo):

$$w(S_s) \leq \sum_{i=0}^{r_s} w(S(i, s)) \tag{6.1}$$

$$\leq \sum_{i=0}^{r_s} w(D|_{W(i, s)}) \tag{6.2}$$

$$\leq w(D) + \sum_{i=0}^{r_s-1} (w(D|_{V_{ik+s-1}}) + w(D|_{V_{ik+s}})) \tag{6.3}$$

porque:

$$(6.1) \quad w_{v, comb(v, a, a')} \leq w_{v, a} + w_{v, a'} \text{ para todos } v \in V(G), a, a' \in L_v,$$

$$(6.2) \quad w(S(i, s)) \leq w(D|_{W(i, s)}) \text{ para todo } i \in \llbracket 0, r_s \rrbracket, \text{ porque } D|_{W(i, s)} \text{ es un coloreo propio en } G[W(i, s)] \text{ para el problema } \Pi',$$

$$(6.3) \quad \text{las capas se superponen en los niveles } ik+s-1 \text{ y } ik+s \text{ para todo } i \in \llbracket 0, r_s-1 \rrbracket.$$

Por el Principio del Palomar,

$$\min \left\{ \sum_{i=0}^{r_s-1} (w(D|_{V_{ik+s-1}}) + w(D|_{V_{ik+s}})) : s \in \llbracket 2, k+1 \rrbracket \right\} \leq \frac{2}{k} w(D).$$

Luego, $\min\{w(S_s) : s \in \llbracket 2, k+1 \rrbracket\} \leq (1 + \frac{2}{k}) w(D)$. En conclusión, podemos aproximar en tiempo polinomial el problema Π por un factor de $1 + \frac{2}{k}$. El siguiente resultado es una consecuencia inmediata de lo anterior.

Teorema 6.3.1. *Sea Π un problema 1-localmente verificable que admite un sistema de vecindades parciales polinomial, con cantidad polinomial de colores, $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{R}^{+\infty}, \leq, +)$ y una función de peso simple, y que además cuenta con una jerarquía de colores. Luego, Π admite un esquema de aproximación de tiempo polinomial (PTAS) en grafos planares.*

6.4. Problemas de maximización

Sea k un entero mayor o igual a 2. Consideremos un problema 1-localmente verificable Π con $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{R}^{-\infty}, \geq, +)$ y que además satisface las restricciones mencionadas en la Sección 6.2 para problemas de maximización. Llamaremos A al conjunto de sus colores neutrales.

Sea G el grafo planar de entrada y sea m la cantidad de niveles de alguna representación plana de G . Al igual que antes, si $m \leq k$ entonces G es k -outerplanar y podemos aplicar directamente el algoritmo del Capítulo 3 para dar una respuesta exacta al problema en tiempo polinomial. Entonces supondremos que $m > k$.

Para una capa $W(i, s)$, definimos la *casi-frontera* $C(i, s)$ como el conjunto de los vértices en los niveles de $W(i, s)$ que son consecutivos a los niveles de la frontera $B(i, s)$. Formalmente,

$$C(i, s) = \begin{cases} V_{s-1} & \text{si } i = 0, \\ V_{(i-1)k+s} \cup V_{ik+s-1} & \text{si } 1 \leq i < r_s, \\ V_{(r_s-1)k+s} & \text{si } i = r_s. \end{cases}$$

Observemos que los niveles de $B(i, s)$ y $C(i, s)$ son precisamente aquellos donde la capa $W(i, s)$ se superpone con $W(i-1, s)$ y $W(i+1, s)$. Dicho de otra manera, $B(i, s) \cup C(i, s) = W(i, s) \cap (W(i-1, s) \cup W(i+1, s))$. Un nivel de la casi-frontera de una capa es uno de los niveles de la frontera de otra capa. Definimos también el conjunto de vértices *profundos* $P(i, s) = I(i, s) - C(i, s)$. La Figura 6.3 ejemplifica estas definiciones.

Para todo $s \in \llbracket 2, k+1 \rrbracket$, notemos $r_s = \lceil \frac{m-s}{k} \rceil$. Para todo $s \in \llbracket 2, k+1 \rrbracket$ y todo $i \in \llbracket 0, r_s \rrbracket$, sea $S(i, s)$ una solución óptima del *problema asociado* Π' , el cual tiene como entrada al grafo $G[W(i, s)]$ y donde además el conjunto L'_v de colores permitidos para cada vértice v se define como

$$L'_v = \begin{cases} A \cap L_v & \text{si } v \in B(i, s) \cup C(i, s), \\ L_v & \text{si no,} \end{cases}$$

su función de verificación $check'$ es tal que:

$$check'(v, c) = \begin{cases} check(v, c) & \text{si } v \in I(i, s), \\ \text{TRUE} & \text{si no,} \end{cases}$$

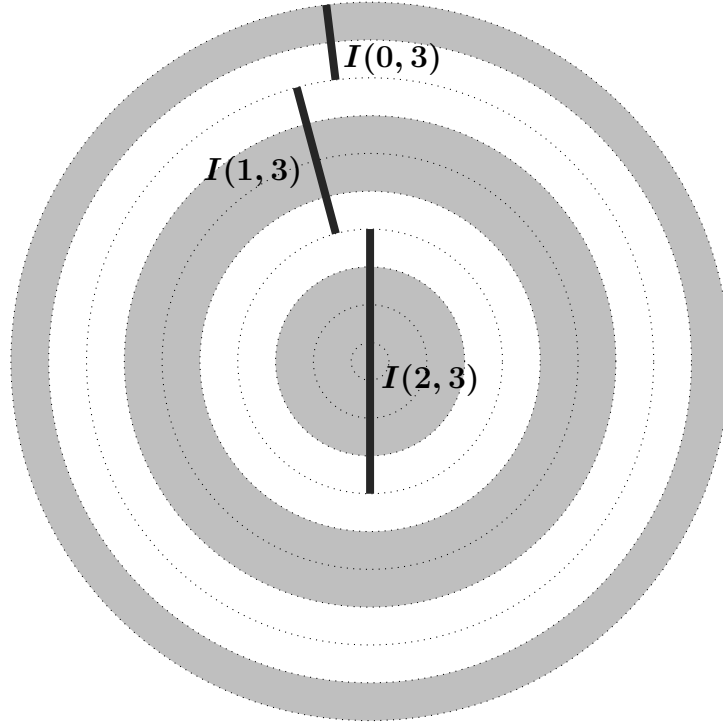


Figura 6.3: Representación de los conjuntos $P(i, s)$ y $C(i, s)$, bajo las mismas condiciones de la Figura 6.2. Los conjuntos de vértices internos $I(0, 3)$, $I(1, 3)$ y $I(2, 3)$ se indican con los segmentos. La casi-frontera $C(0, 3)$ incluye el nivel 2, $C(1, 3)$ los niveles 3 y 6, y $C(2, 3)$ el nivel 7. Los conjuntos $P(0, 3)$, $P(1, 3)$ y $P(2, 3)$ están sombreados en gris.

siendo *check* la función de verificación de Π , y en todo lo demás Π' es análogo a Π . En otras palabras, $S(i, s)$ es un coloreo de costo máximo que cumple la función de verificación de Π para los vértices internos de la capa, pero no necesariamente para los de la frontera, y donde los vértices de la frontera y casi-frontera sólo pueden recibir un color neutral. Es importante mencionar que si existe un coloreo propio c de G para el problema Π , entonces existe cada uno de los coloreos $S(i, s)$. En efecto, por la restricción (2) sobre los colores neutrales, podemos extender $c|_{P(i, s)}$ a todo $W(i, s)$ utilizando colores neutrales. Como $G[W(i, s)]$ es k -outerplanar, podemos determinar $S(i, s)$ en tiempo polinomial.

Para todo $s \in \llbracket 2, k+1 \rrbracket$, sea S_s la unión de todos los coloreos $S(i, s)$ restringidos a $I(i, s)$, con $i \in \llbracket 0, r_s \rrbracket$. Notemos que S_s asigna exactamente un color a cada vértice y que además es un coloreo propio en G (gracias a la restricción (1) sobre los colores neutrales) que se puede obtener en tiempo polinomial. Además, $w(S_s|_{W(i, s)}) = w(S(i, s))$ para todo $i \in \llbracket 0, r_s \rrbracket$, porque $w(S_s|_{I(i, s)}) = w(S(i, s)|_{I(i, s)})$ y los costos de los colores neutrales son todos iguales.

Sea D un coloreo propio óptimo en G . Sea $s \in \llbracket 2, k+1 \rrbracket$. Sea D^s una extensión del coloreo parcial $\cup_{i=0}^{r_s} D|_{P(i, s)}$ a un coloreo propio para el problema Π (sabemos que existe por las restricciones del problema). Para todo $i \in \llbracket 0, r_s \rrbracket$, llamemos $D^{s, i}$ a $D^s|_{W(i, s)}$. Notemos que $D^{s, i}$ es un coloreo propio para el problema Π' aplicado a $G[W(i, s)]$. Por lo tanto, $w(S_s|_{W(i, s)}) = w(S(i, s)) \geq w(D^{s, i})$. En particular, como los costos de los colores neutrales son iguales, tenemos que $w(S_s|_{P(i, s)}) \geq w(D^{s, i}|_{P(i, s)}) = w(D|_{P(i, s)})$.

Entonces:

$$w(S_s) = \sum_{i=0}^{r_s} w(S_s|_{I(i,s)}) \quad (6.4)$$

$$\geq \sum_{i=0}^{r_s} w(S_s|_{P(i,s)}) \quad (6.5)$$

$$\geq \sum_{i=0}^{r_s} w(D|_{P(i,s)}) \quad (6.6)$$

$$= w(D) - \sum_{i=0}^{r_s-1} (w(D|_{V_{ik+s-1}}) + w(D|_{V_{ik+s}})) \quad (6.7)$$

porque:

(6.4) los conjuntos $I(i, s)$ forman una partición de los vértices de G ,

(6.5) $P(i, s) \subseteq I(i, s)$ para todo $i \in \llbracket 0, r_s \rrbracket$, y el costo de asignar colores neutrales es mayor o igual a 0,

(6.6) $w(S_s|_{P(i,s)}) \geq w(D|_{P(i,s)})$ para todo $i \in \llbracket 0, r_s \rrbracket$,

(6.7) los niveles $ik + s - 1$ y $ik + s$, para todo $i \in \llbracket 0, r_s - 1 \rrbracket$, son los únicos que no pertenecen a ningún $P(j, s)$.

Por el Principio del Palomar,

$$\min \left\{ \sum_{i=0}^{r_s-1} (w(D|_{V_{ik+s-1}}) + w(D|_{V_{ik+s}})) : s \in \llbracket 2, k+1 \rrbracket \right\} \leq \frac{2}{k} w(D).$$

Luego, $\max\{w(S_s) : s \in \llbracket 2, k+1 \rrbracket\} \geq (1 - \frac{2}{k}) w(D)$. En conclusión, podemos aproximar en tiempo polinomial el problema Π por un factor de $1 - \frac{2}{k}$. El siguiente resultado es una consecuencia inmediata de lo anterior.

Teorema 6.4.1. *Sea Π un problema 1-localmente verificable que admite un sistema de vecindades parciales polinomial, con cantidad polinomial de colores, $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{R}^{-\infty}, \geq, +)$ y una función de peso simple, y que además cuenta con un conjunto de colores neutrales. Luego, Π admite un esquema de aproximación de tiempo polinomial (PTAS) en grafos planares.*

Capítulo 7

Aplicaciones

En este capítulo mostramos cómo modelar diferentes problemas bajo el marco de problemas localmente verificables, incluyendo propiedades globales en algunos casos. Como consecuencia inmediata de reconocer las restricciones que estos modelos satisfacen, se obtienen sus complejidades parametrizadas por treewidth, clique-width y/o mim-width, según corresponda. Adicionalmente, si satisfacen las restricciones dadas en el Capítulo 6, también se obtiene inmediatamente un esquema de aproximación de tiempo polinomial en grafos planares. A menos que se indique lo contrario, hasta donde sabemos, estos resultados eran desconocidos previo a su publicación en los artículos [16, 41, 103] o a esta tesis. Por otro lado, en las últimas dos secciones veremos cómo modelar problemas ampliamente estudiados en las clases de grafos que son de interés aquí, ya que los problemas así modelados pueden ser fácilmente modificados o combinados con propiedades globales, incluso considerando versiones de distancia de ellos, y además pueden servir de inspiración para el modelo de otros problemas en el marco aquí presentado.

Con el fin de facilitar la lectura, a lo largo de las siguientes secciones supondremos que, a menos que se indique lo contrario, las definiciones de las distintas funciones serán dadas para todos los elementos del dominio.

7.1. Dominación romana

Sea k un entero positivo. Una *función $[k]$ -dominante romana* en un grafo G es una función $f: V(G) \rightarrow \llbracket 0, k+1 \rrbracket$ tal que $\sum_{u \in N_G[v]} f(u) \geq |AN_G^f(v)| + k$ para todo $v \in V(G)$, donde $AN_G^f(v) = \{u \in N_G(v) : f(u) \geq 1\}$ (este conjunto se llama *la vecindad activa de v*). El peso de una función $[k]$ -dominante romana f es $\sum_{v \in V(G)} f(v)$, y el mínimo peso de una tal función en G es el *número de $[k]$ -dominación romana* de G , notado $\gamma_{[k]}(G)$. El problema $[k]$ -DOMINACIÓN ROMANA consiste en calcular el número de $[k]$ -dominación romana de un grafo dado. Este problema fue definido por primera vez en [6] como una generalización de DOMINACIÓN ROMANA y DOMINACIÓN ROMANA DOBLE [21, 62]. En [2] se probó que este segundo problema es NP-difícil para grafos bipartitos y cordales.

Podemos modelar este problema como un problema 1-localmente verificable de la siguiente manera:

- COLORS = $\llbracket 0, k+1 \rrbracket$,

- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w(v, c) = c(v)$,
- $check(v, c) = \left(\sum_{u \in N_G[v]} c(u) \geq |AN_G^c(v)| + k \right)$.

Podemos ver que $\sum_{u \in N_G[v]} c(u) - |AN_G^c(v)| = c(v) + \sum_{u \in AN_G^c(v)} (c(u) - 1)$ y entonces la función *check* es contadora de colores con la siguiente función asociada:

$$check'(v, a, \mu) = \left(a + \sum_{j=1}^{k+1} (j-1)\mu_j \geq k \right).$$

Esta función es además k -estable. Por otro lado, la función de peso también es k -estable y la cantidad de colores es constante. Por los Corolarios 3.1.8, 4.1.11 y 5.1.4, $[k]$ -DOMINACIÓN ROMANA es FPT lineal parametrizado por treewidth, FPT parametrizado por clique-width (lineal en caso de contar con una expresión clique-width) y XP parametrizado por la mim-width de un árbol binario de descomposición dado.

Más aún, notemos que también se cumplen las restricciones del Capítulo 6: es un problema de minimización, con cantidad polinomial de colores, con un sistema de vecindades polinomial, con pesos en los números reales y que admite una jerarquía de colores (en concreto, $comb(v, i, j) = \min(k+1, i+j)$ para todo vértice v y todos $i, j \in \text{COLORS}$). Por lo tanto, admite un PTAS en grafos planares.

Similarmente, basándose en $\{2\}$ -DOMINACIÓN ROMANA [57] (también llamada DOMINACIÓN ITALIANA [112]), en [173] definieron la $\{k\}$ -DOMINACIÓN ROMANA. En este problema hay que encontrar el menor peso de una función $f: [0, k]$ tal que $\sum_{u \in N_G(v)} f(u) \geq k$ para todos los vértices $v \in V(G)$ con $f(v) = 0$. Claramente este problema también es un problema 1-localmente verificable con cantidad constante colores y funciones k -estables, y por lo tanto FPT lineal parametrizado por treewidth, FPT parametrizado por clique-width y XP parametrizado por la mim-width de un árbol binario de descomposición dado. Además también admite un PTAS en grafos planares. Este último hecho resulta interesante ya que $\{2\}$ -DOMINACIÓN ROMANA es NP-difícil en grafos planares [150], incluso en bipartitos planares [81].

Existen muchas otras variantes de $[k]$ -DOMINACIÓN ROMANA para las cuales también se puede dar un modelo con cantidad constante de colores y funciones k -estables, a saber: total [157], independiente [137], independiente exterior [3], perfecta [18] y perfecta doble [75]. Las definiciones formales de todos estos problemas se encuentran en el Apéndice A y la Tabla 7.1 resume sus modelos como problema 1-localmente verificable con funciones contadoras de colores. Estas variantes son, entonces, XP parametrizadas por la mim-width de un árbol binario de descomposición dado, FPT parametrizadas por clique-width y FPT lineal parametrizadas por treewidth. Además, las restricciones para problemas de minimización dadas en el Capítulo 6 se cumplen para las variantes total e independiente exterior, por lo tanto estos dos problemas también admiten PTAS en grafos planares.

Para las variantes con signo [5] y total con signo [172], notemos que podemos modelarlas como problemas con funciones contadoras de colores y entonces, por el Corolario 4.1.9, son XP parametrizadas por clique-width. Sin embargo, no podemos dar un modelo con funciones d -estables para ninguna constante d , por lo tanto su complejidad parametrizada por mim-width sigue abierta.

Variante	COLORS	$check'(v, a, \mu)$
Total	$\llbracket 0, k+1 \rrbracket$	$\left(a + \sum_{j=1}^{k+1} (j-1)\mu_j \geq k \right) \wedge \left(a > 0 \Rightarrow \sum_{j=1}^{k+1} \mu_j \geq 1 \right)$
Independiente	$\llbracket 0, k+1 \rrbracket$	$\left(a + \sum_{j=1}^{k+1} (j-1)\mu_j \geq k \right) \wedge \left(a > 0 \Rightarrow \sum_{j=1}^{k+1} \mu_j = 0 \right)$
Independiente exterior	$\llbracket 0, k+1 \rrbracket$	$\left(a + \sum_{j=1}^{k+1} (j-1)\mu_j \geq k \right) \wedge (a = 0 \Rightarrow \mu_0 = 0)$
Perfecta	$\{0, 1, 2\}$	$a = 0 \Rightarrow \mu_2 = 1$
Perfecta doble	$\{0, 1, 2, 3\}$	$(a = 0 \Rightarrow ((\mu_2 = 2 \wedge \mu_3 = 0) \vee (\mu_2 = 0 \wedge \mu_3 = 1)))$ $\wedge (a = 1 \Rightarrow (\mu_2 = 1 \wedge \mu_3 = 0))$
Con signo	$\{-1, 1, 2\}$	$(a - \mu_{-1} + \mu_1 + 2\mu_2 \geq k) \wedge (a = -1 \Rightarrow \mu_2 \geq 1)$
Total con signo	$\{-1, 1, 2\}$	$(-\mu_{-1} + \mu_1 + 2\mu_2 \geq k) \wedge (a = -1 \Rightarrow \mu_2 \geq 1)$

Tabla 7.1: Algunas variantes de dominación romana. En todos los casos, el conjunto de pesos es $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$ y la función de peso es $w(v, c) = c(v)$. Las funciones de verificación son contadoras de colores y se expresan directamente como tales.

Una variante particular que definimos en [41] es la llamada $[k]$ -DOMINACIÓN ROMANA MAXIMAL, la cual surgió como una generalización de DOMINACIÓN MAXIMAL [131], DOMINACIÓN ROMANA MAXIMAL [1] y DOBLE DOMINACIÓN ROMANA MAXIMAL [4]. Una *función $[k]$ -dominante romana maximal* en un grafo G es una función $f: V(G) \rightarrow \llbracket 0, k+1 \rrbracket$ tal que f es una función $[k]$ -dominante romana y $\{v \in V(G) : f(v) = 0\}$ no es un conjunto dominante. El peso de una tal función f es $\sum_{v \in V(G)} f(v)$, y el mínimo peso de tales funciones en G es el *número de $[k]$ -dominación romana maximal* de G ($\gamma_{m[kR]}(G)$). El problema de determinar $\gamma_{m[kR]}(G)$ puede ser modelado como un problema 1-localmente verificable con $2k+3$ colores, funciones k -estables y con propiedades globales de tamaño:

- $\text{COLORS} = \llbracket -(k+1), k+1 \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w'(v, a, \mu) = |a|$,
- $check'(v, a, \mu) = ((|a| + \sum_{j=1}^{k+1} (j-1)(\mu_j + \mu_{-j}) \geq k) \wedge (a < 0 \Rightarrow \mu_0 = 0))$,
- propiedad global de tamaño: existe por lo menos 1 vértice cuyo color está en $\llbracket -(k-1), -1 \rrbracket$.

En este modelo estamos usando colores especiales (los valores negativos) para aquellos vértices que no son dominados por vértices con color 0, y justamente requerimos que exista al menos un vértice con alguno de estos colores. Por los resultados de los Capítulos 3, 4 y 5, este problema es FPT lineal parametrizado por treewidth, FPT por clique-width y XP por la mim-width de un árbol binario de descomposición dado.

Otra variante que vale la pena mencionar es la llamada $[k]$ -DOMINACIÓN ROMANA GLOBAL que presentamos en [16] como una generalización natural de los problemas DOMINACIÓN ROMANA GLOBAL [153], DOBLE DOMINACIÓN ROMANA GLOBAL [158] y TRIPLE DOMINACIÓN ROMANA GLOBAL [145]. Una *función $[k]$ -dominante romana global* en un grafo G es una función $[k]$ -dominante romana tanto en G como en su

complemento $\overline{G}$. El problema consiste en calcular el mínimo peso de una tal función en un grafo dado.

Vamos a mostrar que este problema es XP parametrizado por clique-width, y para ello primero vamos a definir un problema auxiliar:

[k]-DOMINACIÓN ROMANA GLOBAL DE TAMAÑO ESPECÍFICO

Instancia: Un grafo G y $k + 2$ enteros no negativos $s_0, \dots, s_{k+1}$ tales que $\sum_{i=0}^{k+1} s_i = |V(G)|$.

Pregunta: ¿Es cierto que G admite una función [k]-dominante romana global f tal que, para todo $i \in \llbracket 0, k + 1 \rrbracket$, s_i es igual a la cantidad de vértices $v \in V(G)$ con $f(v) = i$?

Este último problema puede ser modelado como un problema 1-localmente verificable con funciones contadoras de colores y propiedades globales de tamaño y , por el Corolario 4.1.11, es XP parametrizado por clique-width:

- $\text{COLORS} = \llbracket 0, k + 1 \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w'(v, a, \mu) = a$,
- $check'(v, a, \mu) = (a + \sum_{j=1}^{k+1} (j - 1)\mu_j \geq k) \wedge (\sum_{j=1}^{k+1} (j - 1)(s_j - \mu_j) \geq k)$,
- para todo $a \in \text{COLORS}$, pedimos que el tamaño de la clase de color a sea s_a .

Finalmente, para resolver [k]-DOMINACIÓN ROMANA GLOBAL, iteramos sucesivamente sobre todas las posibles combinaciones de valores $s_0, \dots, s_{k+1}$ tales que $\sum_{i=0}^{k+1} s_i = |V(G)|$ y $s_i \geq 0$ para todo $i \in \llbracket 0, k + 1 \rrbracket$. Notemos que la cantidad de tales combinaciones es menor a $(|V(G)| + 1)^{k+2}$. Para cada combinación, resolvemos [k]-DOMINACIÓN ROMANA GLOBAL DE TAMAÑO ESPECÍFICO, y luego buscamos la mejor entre las soluciones encontradas.

7.2. Dominación dual

En el artículo [64] se definen tres variantes recientes del problema de dominación. Dado un grafo particionado en vértices *positivos* (V^+) y *negativos* (V^-), en cada uno de estos tres problemas queremos encontrar un subconjunto de vértices positivos $D \subseteq V^+$ pero que satisfaga distintas condiciones:

- MÁXIMA DOMINACIÓN DUAL ACOTADA (en inglés, *maximum bounded dual domination*, abreviado MBDD): para un entero k dado como entrada, $|N[D] \cap V^-| \leq k$ y $|N[D] \cap V^+|$ es lo más grande posible.
- MÁXIMA DOMINACIÓN DUAL (en inglés, *maximum dual domination*, abreviado MDD): D es tal que maximiza $|N[D] \cap V^+| - |N[D] \cap V^-|$.
- MÍNIMA DOMINACIÓN DUAL NEGATIVA (en inglés, *minimum negative dual domination*, abreviado mNDD): D domina V^+ (es decir, $V^+ \subseteq N[D]$) y minimiza $|N[D] \cap V^-|$.

Todos estos problemas son NP-difíciles y existen algoritmos de tiempo lineal para resolver MDD y mNDD en árboles, así como un algoritmo de tiempo $O(k^2|V(G)|)$ para resolver MBDD en árboles [64].

Vamos a mostrar que todos ellos se pueden modelar como problemas 1-localmente verificables con cantidad constante de colores y funciones 1-estables, uno de ellos adicionalmente con una propiedad global de tamaño. En estos modelos, el conjunto D que buscamos corresponde a la clase de color D .

- MÁXIMA DOMINACIÓN DUAL ACOTADA (MBDD):

- $\text{COLORS} = \{D, N^+, N^-, \bar{N}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$,
- $w'(v, a, \mu) = \begin{cases} 1 & \text{si } a \in \{D, N^+\}, \\ 0 & \text{si no,} \end{cases}$
- $check'(v, a, \mu) = \begin{cases} v \in V^+ & \text{si } a = D, \\ v \in V^+ \wedge \mu_D \geq 1 & \text{si } a = N^+, \\ v \in V^- \wedge \mu_D \geq 1 & \text{si } a = N^-, \\ \mu_D = 0 & \text{si } a = \bar{N}, \end{cases}$
- propiedad global: la cantidad de vértices de color N^- es a lo sumo k .

- MÁXIMA DOMINACIÓN DUAL (MDD):

- $\text{COLORS} = \{D, N^+, N^-, \bar{N}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{Z}^{-\infty}, \geq, +)$,
- $w'(v, a, \mu) = \begin{cases} 1 & \text{si } a \in \{D, N^+\}, \\ -1 & \text{si } a = N^-, \\ 0 & \text{si no,} \end{cases}$
- $check'(v, a, \mu) = \begin{cases} v \in V^+ & \text{si } a = D, \\ v \in V^+ \wedge \mu_D \geq 1 & \text{si } a = N^+, \\ v \in V^- \wedge \mu_D \geq 1 & \text{si } a = N^-, \\ \mu_D = 0 & \text{si } a = \bar{N}. \end{cases}$

- MÍNIMA DOMINACIÓN DUAL NEGATIVA (mNDD):

- $\text{COLORS} = \{D, N^+, N^-, \bar{N}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w'(v, a, \mu) = \begin{cases} 1 & \text{si } a = N^-, \\ 0 & \text{si no,} \end{cases}$
- $check'(v, a, \mu) = \begin{cases} v \in V^+ & \text{si } a = D, \\ v \in V^+ \wedge \mu_D \geq 1 & \text{si } a = N^+, \\ v \in V^- \wedge \mu_D \geq 1 & \text{si } a = N^-, \\ v \in V^- \wedge \mu_D = 0 & \text{si } a = \bar{N}. \end{cases}$

Por lo tanto, estos tres problemas son XP parametrizados por la mim-width de un árbol binario de descomposición dado, y FPT parametrizados por clique-width.

7.3. Coloreos felices

Dado un grafo G y una asignación de colores a vértices c , decimos que un vértice v es *feliz* si todos sus vecinos reciben también el color $c(v)$. El problema llamado MÁXIMO CONJUNTO DE VÉRTICES k -FELICES (en inglés, *k-maximum happy vertices*, abreviado k -MHV), definido en [175], consiste en determinar la máxima cantidad de vértices felices que se pueden obtener extendiendo un coloreo parcial dado, utilizando k colores. Para cada $k \geq 3$, este problema es NP-difícil [175] pero es FPT parametrizado por clique-width y resoluble en tiempo polinomial en grafos de intervalo [31]. En [9] se define una generalización natural a grafos pesados, donde hay una función $w: V(G) \rightarrow \mathbb{N}$ y queremos maximizar la suma de $w(v)$ entre todos los vértices felices v , y se prueba que es FPT parametrizado por treewidth. Podemos modelar k -MHV (tanto en su versión original como pesada) como un problema 1-localmente verificable con $2k$ colores y funciones 1-estables, de la siguiente manera. Sean P el conjunto de vértices pre-coloreados y $p(v)$ el color que recibe cada $v \in P$ en este pre-coloreo. Luego:

- $\text{COLORS} = \{I_1, F_1, \dots, I_k, F_k\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$,
- $w'(v, a, \mu) = \begin{cases} w(v) & \text{si } a = F_i \text{ para algún } i \in [1, k], \\ 0 & \text{si no,} \end{cases}$
- $check'(v, a, \mu) = (v \in P \Rightarrow a \in \{I_{p(v)}, F_{p(v)}\}) \wedge ((\exists i \in [1, k]. a = F_i) \Rightarrow \forall j \in [1, k] - \{i\}. \mu_{I_j} + \mu_{F_j} = 0)$.

En este modelo, los colores indican si un vértice es feliz (F) o no (I). Por tener funciones 1-estables y cantidad constante de colores, es FPT lineal parametrizado por treewidth, FPT parametrizado por clique-width y XP parametrizado por la mim-width de un árbol binario de descomposición dado. Notemos que este es un problema de maximización y el conjunto de colores $\{I_1, \dots, I_k\}$ cumple la definición de los colores neutrales (ver Capítulo 6). Por lo tanto, admite un PTAS en grafos planares.

Un problema relacionado con estos es el de MÁXIMO CONJUNTO FELIZ (en inglés, *maximum happy set*, abreviado MaxHS), definido en [15], donde se busca un subconjunto S de k vértices que maximice la cantidad de vértices $v \in S$ tales que $N[v] \subseteq S$. Este problema es NP-difícil [15] pero también FPT parametrizado por clique-width [142], y resoluble en tiempo polinomial en grafos de intervalo [14]. Con el siguiente modelo mostramos que MaxHS se puede expresar como un problema 1-localmente verificable con 2 colores, funciones 1-estables y una propiedad global de tamaño.

- $\text{COLORS} = \{S, \bar{S}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$,
- $w'(v, a, \mu) = \begin{cases} 1 & \text{si } a = S \text{ y } \mu_{\bar{S}} = 0, \\ 0 & \text{si no,} \end{cases}$
- $check'(v, a, \mu) = \text{TRUE}$,
- propiedad global: el tamaño de la clase de color S es k .

En conclusión, este problema también es XP parametrizado por la mim-width de un árbol binario de descomposición dado.

7.4. Coloreo localmente acotado

Supongamos que tenemos un grafo G , dos enteros positivos p y k , una partición de $V(G)$ en p subconjuntos $V_1, \dots, V_p$, y pk cotas enteras $n_{i,j}$, con $i \in \llbracket 1, p \rrbracket$ y $j \in \llbracket 1, k \rrbracket$, tales que $\sum_{j=1}^k n_{i,j} = |V_i|$ para cada $i \in \llbracket 1, p \rrbracket$. El problema llamado k -COLOREO LOCALMENTE ACOTADO (en inglés, *locally bounded k -coloring*) definido en [25], consiste en determinar si existe un coloreo de G que use k colores y tal que ningún par de vecinos reciba el mismo color y que, para cada $i \in \llbracket 1, p \rrbracket$ y cada $j \in \llbracket 1, k \rrbracket$, la cantidad de vértices con color j en V_i es $n_{i,j}$. Para valores fijos de p y k , este problema es NP-completo [25] y es FPT parametrizado por treewidth y resoluble en tiempo polinomial en cografos [24].

Proponemos el siguiente modelo para este problema, con p y k fijos, como un problema 1-localmente verificable con pk colores, funciones 1-estables y propiedades globales de tamaño.

- $\text{COLORS} = \llbracket 1, p \rrbracket \times \llbracket 1, k \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w'(v, (i, j), \mu) = 0$,
- $check'(v, (i, j), \mu) = (v \in V_i \wedge \sum_{i'=1}^p \mu_{(i', j)} = 0)$,
- propiedad global: para cada par $(i, j) \in \text{COLORS}$, la cantidad de vértices de color (i, j) es $n_{i,j}$.

Por lo tanto, este problema es XP parametrizado por la mim-width de un árbol binario de descomposición dado, y FPT parametrizado por clique-width.

7.5. Coloreos libres de conflictos

Los problemas de *coloreos libres de conflictos* son un conjunto de problemas en los cuales se busca un coloreo del grafo tal que en la vecindad de cada vértice exista un color que aparece exactamente una vez. El primero de estos problemas fue definido en [79] y a partir de entonces diversas variantes fueron estudiadas (por ejemplo, en [30]). Las diferencias son según si la vecindad se considera abierta o cerrada, y según el conjunto de colores. A continuación definiremos algunas que son de nuestro interés para este trabajo.

Dado un grafo G , decimos que un coloreo c de los vértices de G es un:

- *CFON k -coloreo* si $c: V(G) \rightarrow \llbracket 1, k \rrbracket$ y es tal que para cada vértice $v \in V(G)$ existe al menos un vértice $u \in N(v)$ tal que ningún otro vértice en $N(v)$ tiene color $c(u)$.
- *CFCN k -coloreo* si $c: V(G) \rightarrow \llbracket 1, k \rrbracket$ y es tal que para cada vértice $v \in V(G)$ existe al menos un vértice $u \in N[v]$ tal que ningún otro vértice en $N[v]$ tiene color $c(u)$.

- *CFON** *k*-coloreo si $c: V(G) \rightarrow \llbracket 0, k \rrbracket$ y es tal que para cada vértice $v \in V(G)$ existe al menos un vértice $u \in N(v)$ con $c(u) \neq 0$ y tal que ningún otro vértice en $N(v)$ tiene color $c(u)$.
- *CFCN** *k*-coloreo si $c: V(G) \rightarrow \llbracket 0, k \rrbracket$ y es tal que para cada vértice $v \in V(G)$ existe al menos un vértice $u \in N[v]$ con $c(u) \neq 0$ y tal que ningún otro vértice en $N[v]$ tiene color $c(u)$.

Cada problema, entonces, consiste en determinar si existe un tal coloreo. Los problemas sobre coloreos CFCN* y CFON* son NP-completos [7] y además fue probado en [30] que estos problemas son FPT parametrizados por clique-width más k . Más aún, también en [30], muestran que todo grafo de intervalo admite un CFON* 3-coloreo y preguntan si existe un algoritmo polinomial para resolver el problema de coloreo CFON* en grafos de intervalo.

Mostraremos cómo modelar uno de ellos, CFON* *k*-coloreo, como un problema 1-localmente verificable con $k + 1$ colores y funciones 2-estables. Los restantes problemas también se pueden modelar de forma muy similar como problemas 1-localmente verificables con funciones 2-estables y cantidad constante de colores.

- $\text{COLORS} = \{0, \dots, k\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w'(v, a, \mu) = 0$,
- $check'(v, a, \mu) = (\exists j \in \llbracket 1, k \rrbracket. \mu_j = 1)$.

Como consecuencia, todos estos problemas son XP parametrizados por $w + k$, donde w es la mim-width de un árbol binario de descomposición dado, y FPT parametrizados por clique-width más k . Como los grafos de intervalo tienen mim-width 1, esto responde a la pregunta planteada en [30].

Independientemente, en [26] se presentan fórmulas en lógica A&C DN para dos de estas variantes, probando así también que estos problemas son XP parametrizados por $w + k$, donde w es la mim-width de un árbol binario de descomposición dado. Sin embargo, la complejidad que resulta de aplicar el algoritmo de [26], $n^{O(wk^2)}$, es ligeramente mayor que la resultante de aplicar el algoritmo de la Sección 5.1, $n^{O(wk)}$.

7.6. Empaquetamientos generalizados

En [113] definen la siguiente generalización de los problemas de empaquetamiento. Sea G un grafo y sean k y u dos vectores de enteros positivos indexados por los vértices de G . Una *función* (k, u) -empaquetadora de G es una función $f: V(G) \rightarrow \mathbb{N}$ tal que $f(v) \leq u[v]$ y $\sum_{w \in N_G[v]} f(w) \leq k[v]$ para todo $v \in V(G)$. El problema de la FUNCIÓN EMPAQUETADORA GENERALIZADA (GPF) tiene como entrada a G , k y u , y pide encontrar el mayor peso de una función (k, u) -empaquetadora. Cuando contamos con la restricción adicional de que $k[v] \leq M$ para todo $v \in V(G)$, para algún entero positivo M fijo, hablamos del problema M -GPF. Fue probado en [113] que, para toda constante entera positiva M , el problema M -GPF se puede resolver en tiempo polinomial en grafos de clique-width acotada, y que GPF se puede resolver en tiempo lineal

en grafos llamados fuertemente cordales (en inglés, *strongly chordal*). Sin embargo, la complejidad de GPF en grafos de clique-width acotada permanecía abierta.

Veamos que GPF puede ser modelado como un problema 1-localmente verificable con $U = \max\{u[v] : v \in V(G)\}$ colores:

- $\text{COLORS} = \llbracket 0, U \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$,
- $w(v, c) = c(v)$,
- $check(v, c) = \left(c(v) \leq u[v] \wedge \sum_{w \in N_G[v]} f(w) \leq k[v] \right)$.

Este modelo admite un sistema de vecindades parciales universal. En efecto, sea $T = \max_{v \in V(G)} \{ \min(k[v] + 1, \sum_{w \in N(v)} u[w]) \}$, luego:

- $\mathcal{N} = \llbracket 0, T \rrbracket$,
- $n \boxplus n' = \min(T, n + n')$,
- $newN(w, b) = \min(T, b)$,
- $w_{v,a}(n) = a$,
- $check_{v,a}(n) = (a \leq u[v] \wedge n + a \leq k[v])$.

Es más, este sistema es polinomial (resp. constante) si U y T son polinomiales (resp. constantes) en $|V(G)|$. Por lo tanto, si $k[v]$ y $u[v]$ son polinomiales (resp. constantes) en $|V(G)|$ para todo $v \in V(G)$, entonces GPF es XP (resp. FPT) parametrizado por clique-width.

En el caso de M -GPF, notemos que la cantidad de colores es a lo sumo $M + 1$ y además las funciones de peso y verificación son M -estables:

- $w'(v, a, \mu) = a$,
- $check'(v, a, \mu) = \left(a \leq u[v] \wedge a + \sum_{i=0}^M i \cdot \mu_i \leq k[v] \right)$.

Entonces, para toda constante entera positiva M , el problema M -GPF es además XP parametrizado por la mim-width de un árbol binario de descomposición dado.

7.7. Problemas de partición en subgrafos densos

7.7.1. Comunidades

Dado un grafo G , una *estructura comunitaria* (en inglés, *community structure*, definida en [146]) es una partición $\{C_1, \dots, C_k\}$ de los vértices de G en $k \geq 2$ conjuntos llamados *comunidades* y tales que

- $|C_i| \geq 2$ para todo $i \in \llbracket 1, k \rrbracket$, y
- $\frac{|N_G(v) \cap C_i|}{|C_i| - 1} \geq \frac{|N_G(v) \cap C_j|}{|C_j|}$ para todos $i, j \in \llbracket 1, k \rrbracket$, con $i \neq j$, y $v \in C_i$.

El problema de decidir si un grafo dado admite una estructura comunitaria con k comunidades se conoce como el problema de k -COMUNIDAD, y fue originalmente definido para $k = 2$ en [19]. La complejidad de 2-COMUNIDAD todavía es una pregunta abierta, conociéndose sólo unas pocas clases de grafos en las cuales este problema es resoluble en tiempo polinomial. Estas clases incluyen grafos de grado máximo 3 y grafos de grado mínimo $|V(G)| - 3$ [19].

Mostraremos aquí que, para cada valor fijo de k , k -COMUNIDAD es XP parametrizado por clique-width. Utilizaremos un enfoque muy similar al de $[k]$ -DOMINACIÓN ROMANA GLOBAL: primero definiremos una variante del problema en la cual requerimos un tamaño fijo para cada comunidad, para luego iterar sobre todos los posibles tamaños.

k -COMUNIDAD DE TAMAÑO ESPECÍFICO

Instancia: Un grafo G y k enteros $s_1, \dots, s_k \geq 2$ tales que $\sum_{i=1}^k s_i = |V(G)|$.

Pregunta: ¿Es cierto que G admite una estructura comunitaria con k comunidades $\{C_1, \dots, C_k\}$ tales que $|C_i| = s_i$ para todo $i \in \llbracket 1, k \rrbracket$?

Este último problema puede ser modelado como un problema 1-localmente verificable con k colores, funciones contadoras de colores y propiedades globales de tamaño, y por lo tanto es XP parametrizado por clique-width más k .

- $\text{COLORS} = \llbracket 1, k \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w'(v, a, \mu) = 0$,
- $check'(v, a, \mu) = \left(\forall b \in \llbracket 1, k \rrbracket. \frac{\mu_a}{s_a - 1} \geq \frac{\mu_b}{s_b} \right)$,
- propiedad global: para todo $a \in \text{COLORS}$, el tamaño de la clase de color a es s_a .

Luego, para resolver k -COMUNIDAD podemos iterar sobre todas las posibles combinaciones de $s_1, \dots, s_k$ tales que $\sum_{i=1}^k s_i = |V(G)|$ y $s_i \geq 2$ para todo $i \in \llbracket 1, k \rrbracket$, resolviendo k -COMUNIDAD DE TAMAÑO ESPECÍFICO para cada una de ellas. Por lo tanto, también es XP parametrizado por clique-width para cada valor fijo de k .

Un análisis muy similar se puede hacer para la variante k -COMUNIDAD DÉBIL, definida en [19], en la cual se requiere que la desigualdad a cumplir por cada vértice v sea $\frac{|N_G[v] \cap C_i|}{|C_i| - 1} \geq \frac{|N_G(v) \cap C_j|}{|C_j|}$ para todos $i, j \in \llbracket 1, k \rrbracket$, con $i \neq j$ y $v \in C_i$. Es evidente que alcanza con tomar la misma solución que antes pero cambiando la definición de $check'$ por $check'(v, a, \mu) = \left(\forall b \in \llbracket 1, k \rrbracket. \frac{\mu_a + 1}{s_a - 1} \geq \frac{\mu_b}{s_b} \right)$. Así, el problema k -COMUNIDAD DÉBIL es XP parametrizado por clique-width más k .

Otras variantes que podemos considerar son k -COMUNIDAD BALANCEADA (DÉBIL), las cuales consisten en determinar si un grafo admite una estructura comunitaria (débil) con k comunidades del mismo tamaño. Claramente, estos problemas son equivalentes a k -COMUNIDAD (DÉBIL) DE TAMAÑO ESPECÍFICO con $s_i = \frac{|V(G)|}{k}$ para todo $i \in \llbracket 1, k \rrbracket$, y así también son XP parametrizados por clique-width más k . Lo interesante de estas variantes es que se pudo probar que 2-COMUNIDAD BALANCEADA (DÉBIL) son NP-completos en general [19, 77] y además se había estudiado un algoritmo polinomial para resolverlos en grafos de treewidth acotado [19].

7.7.2. PDS

Sean G un grafo y $S \subset V(G)$ tal que $2 \leq |S| < |V(G)|$. Decimos que el subgrafo de G inducido por S ($G[S]$) es *proporcionalmente denso* si $\frac{|N_G(v) \cap S|}{|S|-1} \geq \frac{|N_G(v) \cap \bar{S}|}{|\bar{S}|}$ para todo $v \in S$. El problema llamado MÁXIMO SUBGRAFO PROPORCIONALMENTE DENSO (en inglés, *maximum proportionally dense subgraph*, abreviado MAX-PDS), definido en [20], consiste en encontrar un subgrafo del grafo de entrada que sea proporcionalmente denso y de tamaño máximo. Este problema es NP-difícil incluso en grafos split o en grafos bipartitos, y es resoluble en tiempo lineal en grafos hamiltonianos cúbicos (dado un ciclo hamiltoniano) [20].

Empleando la misma técnica que para los problemas de comunidades, definimos:

MAX-PDS DE TAMAÑO ESPECÍFICO

Instancia: Un grafo G y un entero s tal que $2 \leq s < |V(G)|$.

Pregunta: ¿Es cierto que G admite un subgrafo proporcionalmente denso de s vértices?

Y lo modelamos como un problema 1-localmente verificable con 2 colores, funciones contadoras de colores y propiedades globales de tamaño:

- $\text{COLORS} = \{s, \bar{s}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w'(v, a, \mu) = 0$,
- $check'(v, a, \mu) = \left(a = s \Rightarrow \frac{\mu_s}{s_s-1} \geq \frac{\mu_{\bar{s}}}{s_{\bar{s}}} \right)$, siendo $s_{\bar{s}} = |V(G)| - s_s$,
- propiedad global: el tamaño de la clase de color s es s .

Por lo tanto este problema y MAX-PDS son XP parametrizados por clique-width.

Una variante de este problema, también definida en [20], es la llamada EXTENSIÓN DE UN SUBGRAFO PROPORCIONALMENTE DENSO (PDS Extension). En ella, dado un subconjunto de vértices U , buscamos un subgrafo proporcionalmente denso $G[S]$ tal que $U \subset S$. Por supuesto este problema también es NP-difícil pero hasta el momento no se conocían algoritmos polinomiales para ninguna clase de grafos. Simplemente modificando el modelo anterior de modo que $L_v = \{s\}$ para todo $v \in U$, obtenemos el primer algoritmo polinomial para clases de grafos de clique-width acotada.

7.7.3. Quasi-cliques

Dado un grafo G y un número real $\gamma \in (0, 1]$, una γ -quasi-clique basada en grados, definida en [139] bajo el nombre de *grafo γ -quasi-completo*, es un subconjunto $S \subseteq V(G)$ tal que el grado de cada vértice en el subgrafo $G[S]$ es al menos $\gamma(|S| - 1)$, es decir, tal que $\frac{|N_G(v) \cap S|}{|S|-1} \geq \gamma$ para todo $v \in S$. El problema MÁXIMA γ -QUASI-CLIQUE BASADA EN GRADOS consiste en encontrar una γ -quasi-clique basada en grados de tamaño máximo en un grafo dado. Este problema es NP-difícil para cualquier γ fijo [148]. Notemos que procediendo de forma prácticamente idéntica al caso de Max-PDS, pero cambiando la función $check'$ por $check'(v, a, \mu) = \left(a = s \Rightarrow \frac{\mu_s}{s_s-1} \geq \gamma \right)$, obtenemos que este problema es XP parametrizado por clique-width.

La *densidad* de un grafo H se define como $\frac{2|E(H)|}{|V(H)|(|V(H)|-1)}$. Dado un grafo G y un número real $\gamma \in (0, 1]$, una γ -quasi-clique (también llamada γ -quasi-clique basada en aristas) es un subconjunto $S \subseteq V(G)$ tal que la densidad de $G[S]$ es al menos γ . Cuando $\gamma = 1$, una γ -quasi-clique no es más que una clique. El problema de encontrar una γ -quasi-clique máxima en un grafo dado es NP-difícil para cualquier $\gamma \in (0, 1]$ [149]. Para resolver este problema en tiempo polinomial en grafos de clique-width acotada no alcanza con modificar solamente la función *check* del enfoque anterior, tenemos que fijar de alguna manera la cantidad de aristas de la quasi-clique. Esto se puede hacer agregando una propiedad global de composición de condiciones locales que asegure que la cantidad de aristas entre vértices de color s sea algún valor prefijado. Entonces, para cada $s_s \in \llbracket 1, |V(G)| \rrbracket$ y $m_s \in \llbracket 1, |E(G)| \rrbracket$ tales que $\frac{2m_s}{s_s(s_s-1)} \geq \gamma$, resolvemos el siguiente problema 1-localmente verificable con propiedades globales que admite un sistema de vecindades parciales universal polinomial:

- $\text{COLORS} = \{s, \bar{s}\}$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w(v, c) = 0$,
- $check(v, c) = \text{TRUE}$,
- propiedad global 1: la cantidad de vértices de color s es s_s (expresada como composición de condiciones locales tenemos: $M^1 = \llbracket 0, s_s \rrbracket$, $m \otimes^1 m' = \text{mín}(m + m', s_s)$, $X^1 = \{s_s\}$, $f^1(v, c) = 1$ si $c(v) = s$ y $f^1(v, c) = 0$ si no),
- propiedad global 2: la cantidad de aristas entre vértices de color s es m_s (expresada como composición de condiciones locales tenemos: $M^2 = \llbracket 0, 2m_s \rrbracket$, $m \otimes^2 m' = \text{mín}(m + m', 2m_s)$, $X^2 = \{2m_s\}$, $f^2(v, c) = |\{u \in N(v) : c(u) = s\}|$ si $c(v) = s$ y $f^2(v, c) = 0$ si no),
- $\mathcal{N} = \llbracket 0, |\Delta(G)| \rrbracket$,
- $n \boxplus n' = \text{mín}(n + n', |\Delta(G)|)$,
- $newN(u, b) = \begin{cases} 1 & \text{si } b = s \\ 0 & \text{si no,} \end{cases}$
- $w_{v,a}(n) = 0$,
- $check_{v,a}(n) = \text{TRUE}$,
- $f_{v,a}^1(n) = \begin{cases} 1 & \text{si } a = s \\ 0 & \text{si no,} \end{cases}$
- $f_{v,a}^2(n) = \begin{cases} n & \text{si } a = s \\ 0 & \text{si no,} \end{cases}$

Entonces cada uno de estos problemas es XP parametrizado por clique-width. Luego basta con buscar la mejor solución entre todas las obtenidas, es decir, una solución de peso 0 que maximice s_s . Por lo tanto, el problema de encontrar el máximo tamaño de una γ -quasi-clique es XP parametrizado por clique-width. Observemos que si bien las funciones *check*, w y f^1 son 1-estables, la función f^2 es no-estable, y por este motivo no podemos aplicar el algoritmo del Capítulo 5.

7.8. b-coloreo

Dado un grafo G , un *b-coloreo* de G es un coloreo de los vértices de G de modo que ningún par de vecinos reciba el mismo color y que cada clase de color contenga por lo menos un vértice (llamado *b-vértice*) que tenga vecinos en todas las otras clases de colores. El *número b-cromático* de G es el máximo k tal que G admite un b-coloreo con k colores. Estas nociones fueron definidas en [118]. En el problema conocido como B-COLOREO, dado un grafo G y un entero k , se pregunta si G admite un b-coloreo con k colores. Este problema es NP-completo incluso en grafos bipartitos [129]. Sin embargo, este problema es XP parametrizado por clique-width [122] y XP parametrizado por *thinness propia* (*pthin*) más k [39].

Mostraremos aquí que B-COLOREO puede ser modelado como un problema 1-localmente verificable con $2k$ colores, con funciones 1-estables y propiedades globales de tamaño:

- $\text{COLORS} = \llbracket -k, -1 \rrbracket \cup \llbracket 1, k \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\{0, 1\}, \leq, \text{máx})$,
- $w'(v, a, \mu) = 0$,
- $check'(v, a, \mu) = \begin{cases} \mu_a + \mu_{-a} = 0 & \text{si } a > 0, \\ \mu_a + \mu_{-a} = 0 \wedge \forall i \in \llbracket 1, k \rrbracket - \{a\} . \mu_i + \mu_{-i} \geq 1 & \text{si no,} \end{cases}$
- propiedad global: para cada $a \in \llbracket -k, -1 \rrbracket$, el tamaño de la clase de color a es al menos 1.

Es fácil ver que un coloreo propio que satisface la propiedad global se corresponde con un b-coloreo, donde los vértices cuyo color es un número negativo representan a los b-vértices. De esta manera, B-COLOREO es XP parametrizado por $w + k$, siendo w la *mim-width* de un árbol binario de descomposición dado, y FPT parametrizado por clique-width más k . Como $mimw(G) \leq pthin(G)$ (consecuencia directa de [39, Teorema 9]), este resultado generaliza el obtenido en [39] respecto a B-COLOREO en grafos de *thinness propia* acotada.

Tal como ocurrió con los problemas de coloreos libres de conflictos, simultáneamente en [26] se expresa este problema en lógica A&C DN, para k fijo. Nuevamente, la complejidad que resulta de aplicar el algoritmo de [26] a esta expresión en lógica A&C DN es ligeramente mayor que la resultante de aplicar el algoritmo de la Sección 5.1 a nuestra formulación como problema 1-localmente verificable con funciones 1-estables.

7.9. Coloreo aditivo

Un *coloreo aditivo* (también llamado *etiquetado afortunado*), definido en [70], es una asignación de enteros positivos a vértices de un grafo tal que para cada par de vecinos, las sumas de los valores asignados a sus vecinos son diferentes. El menor valor posible del máximo entero asignado a los vértices de un grafo G se denota $\eta(G)$. Se demostró en [11] que $\eta(G) \leq \Delta(G)^2 - \Delta(G) + 1$ para todo grafo G . Llamemos $C(G)$ a esta cota superior.

Vamos a modelar el problema de calcular $\eta(G)$ como un problema 1-localmente verificable con un sistema de vecindades parciales polinomial. Los colores son pares (n, s) , donde n representa el número asignado al vértice y s la suma esperada de los números asignados a los vecinos.

- $\text{COLORS} = \llbracket 1, C(G) \rrbracket \times \llbracket 1, \Delta(G)C(G) \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, \text{máx})$,
- $\text{W}(v, c) = c(v)_1$,
- $\text{check}(v, c) = (\forall u \in N_G(v). c(u)_2 \neq c(v)_2) \wedge \left(c(v)_2 = \sum_{u \in N_G(v)} c(u)_1 \right)$,
- $\mathcal{N}_{v,i} = \text{BOOL} \times \llbracket 1, \Delta(G)C(G) + 1 \rrbracket$,
- $n \boxplus^{v,i} n' = (n_1 \wedge n'_1, \min(n_2 + n'_2, \Delta(G)C(G) + 1))$,
- $\text{new}N_{v,i}(u, j) = (i_2 \neq j_2, j_1)$,
- $\text{W}_{v,i}(n) = i_1$,
- $\text{check}_{v,i}(n) = (n_1 \wedge (i_2 = n_2))$.

La cantidad de colores es $O(\Delta(G)^5)$ y $\mathcal{N}$ es $O(\Delta(G)^3)$, lo que implica que este problema es XP parametrizado por treewidth, y FPT lineal parametrizado por carvingwidth. Otro algoritmo polinomial en clases de treewidth acotada fue obtenido por R. Grappe, L. N. Grippo y M. Valencia-Pabon (comunicación personal).

7.10. Mínima violación cromática

El presente problema NP-difícil fue definido en [44] como una generalización del problema de k -coloreo. Dado un grafo G , un conjunto de *aristas débiles* $F \subseteq E(G)$ y un entero positivo k , el problema MÍNIMA VIOLACIÓN CROMÁTICA pregunta si existe un k -coloreo del grafo $G' = (V(G), E(G) - F)$ que minimiza la cantidad de aristas débiles de G con ambos extremos del mismo color. Podemos reducir este problema al siguiente problema 1-localmente verificable en el grafo de subdivisión $S(G)$:

- $L_v = \llbracket 1, k \rrbracket$ para todo vértice $v \in V(G)$,
 $L_{uv} = L_u \times L_v$ para toda arista $uv \in E(G)$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,

- para todo vértice $v \in V(G)$, $w(v, c) = 0$,
para toda arista $uv \in E(G)$, $w(uv, c) = 1$ si $c(uv) = (i, i)$ para algún $i \in L_u \cap L_v$,
y $w(uv, c) = 0$ si no,
- $check(v, c) = \text{TRUE}$ para todo vértice $v \in V(G)$,
 $check(uv, c) = (c(uv) = (c(u), c(v)))$ para toda arista débil $uv \in F$, y
 $check(uv, c) = (c(uv) = (c(u), c(v)) \wedge c(u) \neq c(v))$ para toda arista no-débil
 $uv \in E(G) - F$.

Esencialmente, las aristas de G son coloreadas con pares de enteros (representando los colores de los extremos), los pesos son tales que si una arista recibe un color que es un par de números iguales entonces su peso es 1 y de lo contrario es 0, y la función de verificación controla que efectivamente los enteros del par se correspondan con los colores de los extremos y además controla que para las aristas no-débiles (fuera de F) estos enteros sean distintos.

A continuación proponemos un sistema de vecindades parciales constante para el modelo anterior:

- $\mathcal{N}_{v,i} = \text{BOOL}$ para todos $v \in V(S(G))$, $i \in L_v$,
- $n \boxplus^{v,i} n' = (n \wedge n')$ para todos $v \in V(S(G))$, $i \in L_v$ y $n, n' \in \mathcal{N}_{v,i}$,
- $newN_{v,i}(e, j) = \text{TRUE}$ para todos $v \in V(G)$, $i \in L_v$, $e \in N_{S(G)}(v)$, $j \in L_e$,
 $newN_{uv,(c_u,c_v)}(u, j) = (j = c_u)$ para todos $uv \in E(G)$, $c_u \in L_u$, $c_v \in L_v$, $j \in L_u$,
 $newN_{uv,(c_u,c_v)}(v, j) = (j = c_v)$ para todos $uv \in E(G)$, $c_u \in L_u$, $c_v \in L_v$, $j \in L_v$,
- $w_{v,i}(n) = 0$ para todos $v \in V(G)$, $i \in L_v$,
 $w_{uv,(i,i)}(n) = 1$ para todos $uv \in E(G)$, $i \in L_u \cap L_v$,
 $w_{uv,(i,j)}(n) = 0$ para todos $uv \in E(G)$, $i \in L_u$, $j \in L_v - \{i\}$,
- $check_{v,i}(n) = \text{TRUE}$ para todos $v \in V(G)$, $i \in L_v$, $n \in \mathcal{N}_{v,i}$,
 $check_{uv,(i,j)}(n) = n$ para todos $uv \in F$, $i \in L_u$, $j \in L_v$, $n \in \mathcal{N}_{uv,(i,j)}$,
 $check_{uv,(i,j)}(n) = (n \wedge i \neq j)$ para todos $uv \in E(G) - F$, $i \in L_u$, $j \in L_v$, $n \in \mathcal{N}_{uv,(i,j)}$.

Por lo tanto, MÍNIMA VIOLACIÓN CROMÁTICA es XP parametrizado por treewidth. Más aún, es FPT lineal parametrizado por treewidth más k .

7.11. Dominación Grundy

Una sucesión $S = (v_1, \dots, v_k)$ de vértices distintos de un grafo G es una *sucesión dominante* si $\{v_1, \dots, v_k\}$ es un conjunto dominante de G . Por otro lado, S es una *sucesión legal* si $N[v_i] - \bigcup_{j=1}^{i-1} N[v_j] \neq \emptyset$ for each i , y es *legal dominante* si es legal y dominante. Decimos que v_i *marca* a los vértices del conjunto $N[v_i] - \bigcup_{j=1}^{i-1} N[v_j]$, y que v_i es el *marcador* de cada $u \in N[v_i] - \bigcup_{j=1}^{i-1} N[v_j]$ (notemos que cada vértice en $V(G)$ tiene un único marcador).

El problema de DOMINACIÓN GRUNDY, definido en [46], consiste en determinar la máxima longitud de una sucesión legal dominante en un grafo dado. En ese mismo artículo se prueba que es NP-difícil incluso en grafos cordales. Por otra parte, se puede

resolver en tiempo polinomial en árboles, grafos split y en cografos [46], y en grafos block [102].

Dada una sucesión legal S , a cada vértice $v \in V(G)$ se le puede asociar un par (p_v, f_v) , donde p_v es la posición de v en S (o “ $\perp$ ” si v no está en S) y f_v indica la posición en S del marcador de v , si existe, o cualquier posición en caso contrario. Directamente de la definición de sucesiones legales, podemos deducir el siguiente enunciado. Un conjunto $\{(p_v, f_v) : v \in V(G)\}$ determina a una sucesión legal si y sólo si:

1. para todo $v \in V(G)$, existe a lo sumo un $u \in N_G[v]$ tal que $f_v = p_u$, y $f_v \leq p_w$ para todo $w \in N_G[v]$ (i.e., v es marcado correctamente o no es marcado),
2. para todo $v \in V(G)$, si $p_v \neq \perp$ entonces existe $u \in N_G[v]$ tal que $f_u = p_v$ (i.e., si v aparece en la sucesión entonces marca a al menos un vértice), y
3. $p_v \neq p_u$ para todo $u, v \in V(G)$ tal que $u \neq v$ y $p_u \neq \perp$ (i.e., dos vértices que aparecen en la sucesión no pueden tener la misma posición en ella).

Notemos que las condiciones 1 y 2 son localmente verificables, mientras que la condición 3 no lo es. Sin embargo, afirmamos que DOMINACIÓN GRUNDY puede ser reducido al siguiente problema 1-localmente verificable. Sean $n = |V(G)|$ y $\perp = n + 1$.

- $\text{COLORS} = ([1, n] \cup \{\perp\}) \times [1, n]$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$,
- $w(v, c) = \begin{cases} 0 & \text{si } c(v)_1 = \perp, \\ 1 & \text{si no,} \end{cases}$
- $check(v, c) = ((\text{existe a lo sumo un } u \in N_G[v] \text{ tal que } c(v)_2 = c(u)_1) \wedge (\forall u \in N_G[v]. c(v)_2 \leq c(u)_1) \wedge (c(v)_1 \neq \perp \Rightarrow \exists u \in N_G[v]. c(v)_1 = c(u)_2) \wedge (c(v)_1 \neq \perp \Rightarrow \forall u \in N_G(v). c(v)_1 \neq c(u)_1)).$

En efecto, sean G un grafo y c un coloreo propio de G en el anterior problema 1-localmente verificable. Sean $(p_v, f_v) = c(v)$ para todo $v \in V(G)$. Las condiciones 1 y 2 son trivialmente satisfechas. Supongamos que no se cumple la condición 3. Luego, existen dos vértices distintos u, v tales que $p_u = p_v \neq \perp$. Por definición de $check$, sabemos que

- $u \notin N_G[v]$,
- si existe un vértice w tal que $f_w > p_v = p_u$ entonces $v \notin N_G[w]$ y $u \notin N_G[w]$ (así, $w \notin N_G[v]$ y $w \notin N_G[u]$), y
- si existe un vértice $w \in N_G[u]$ tal que $f_w = p_v = p_u$ entonces $w \notin N_G[v]$.

Luego, $N_G[u] \cap N_G[v] = F$ tal que $f_w < p_u = p_v$ para todo $w \in F$. Ahora podemos asignar colores (p'_w, f'_w) para cada $w \in V(G)$, de modo que

$$p'_w = \begin{cases} p_w & \text{si } p_w \leq p_u \text{ y } w \neq v \\ p_w + 1 & \text{si no} \end{cases}$$

y

$$f'_w = \begin{cases} f_w & \text{si } f_w < p_u, \text{ o } f_w = p_u \text{ y } w \notin N_G[v] \\ f_w + 1 & \text{si no.} \end{cases}$$

Es decir, avanzamos una posición a v y a todos los vértices que aparecen después de u en S y aumentamos los f_w necesarios. Es fácil ver que esta nueva asignación preserva la “legalidad” de u (i.e., si $N_G[u] - \{z : z \in N_G[w] \wedge p_w < p_u\} \neq \emptyset$ entonces $N_G[u] - \{z : z \in N_G[w] \wedge p'_w < p'_v\} \neq \emptyset$) y también la de todos los otros vértices.

Ya que la cantidad de colores es $O(|V(G)|^2)$ y con el fin de dar un algoritmo XP parametrizado por treewidth para DOMINACIÓN GRUNDY, definimos el siguiente sistema de vecindades parciales constante:

- $\mathcal{N}_{v,(p,f)} = \{0, 1, 2\} \times \text{BOOL} \times \text{BOOL} \times \text{BOOL}$,
- $n \boxplus^{v,(p,f)} n' = (\min(n_1 + n'_1, 2), n_2 \wedge n'_2, n_3 \vee n'_3, n_4 \wedge n'_4)$,
- $\text{new}N_{v,(p_v,f_v)}(u, (p_u, f_u)) = n$ donde:
 - $n_1 = \begin{cases} 1 & \text{si } f_v = p_u \\ 0 & \text{si no,} \end{cases}$
 - $n_2 = (f_v \leq p_u)$,
 - $n_3 = (p_v = f_u)$,
 - $n_4 = (p_v \neq p_u)$,
- $w_{v,(p,f)}(n) = \begin{cases} 0 & \text{si } p = \perp \\ 1 & \text{si no,} \end{cases}$
- $\text{check}_{v,(p,f)}(n) = ((n_1 < 2 \wedge (n_1 = 1 \Rightarrow f \neq p)) \wedge (n_2 \wedge f \leq p) \wedge (p \neq \perp \Rightarrow (n_3 \vee p = f)) \wedge (p \neq \perp \Rightarrow n_4))$.

De forma muy similar también podemos probar que la variante *total* de este problema (DOMINACIÓN TOTAL GRUNDY [47], donde no se permite que ningún vértice se marque a sí mismo) es XP parametrizada por treewidth.

Ambos son problemas de maximización donde el conjunto de colores $\{(\perp, f) : f \in [1, n]\}$ cumple la definición de los colores neutrales (ver Capítulo 6). Por lo tanto, cada uno admite un PTAS en grafos planares.

7.12. Problemas que involucran aristas

Consideremos problemas localmente verificables donde cada arista debe cumplir con una determinada condición respecto a sus extremos y aristas adyacentes. Mostraremos cómo reducir estos problemas a problemas 1-localmente verificables en el grafo dentado $J(G)$ del grafo G de entrada. Consideramos separadamente dos tipos de problemas: aquellos en los que no hay requerimientos sobre las aristas que involucren a las aristas adyacentes, y aquellos en los que sí.

Para el primer tipo de problemas, la reducción es directa. Quizás necesitemos duplicar los colores para distinguir los colores asignados a los vértices de los colores asignados a las aristas. Como ejemplo, vemos cómo modelar CUBRIMIENTO POR VÉRTICES (también llamado “cobertura de vértices”, *vertex cover* en inglés) como un problema 1-localmente verificable con un sistema de vecindades parciales constante.

- $L_v = \{0, 1\}$ para todo vértice $v \in V(G)$ y
 $L_{uv} = \{0\}$ para toda arista $uv \in E(G)$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$,
- $w(\nu, c) = c(\nu)$ para todo elemento $\nu \in V(J(G))$,
- $check(v, c) = \text{TRUE}$ para todo vértice $v \in V(G)$ y
 $check(uv, c) = (c(u) + c(v) \geq 1)$ para toda arista $uv \in E(G)$,
- $\mathcal{N}_{\nu,i} = \{0, 1\}$ para todos $\nu \in V(J(G))$, $i \in L_\nu$,
- $n \boxplus^{\nu,i} n' = \min(n + n', 1)$ para todos $\nu \in V(J(G))$, $i \in L_\nu$,
- $newN_{\nu,i}(\nu', j) = j$ para todos $\nu, \nu' \in V(J(G))$, $i \in L_\nu$, $j \in L_{\nu'}$,
- $w_{\nu,i}(n) = i$ para todos $\nu \in V(J(G))$, $i \in L_\nu$,
- $check_{v,i}(n) = \text{TRUE}$ para todo vértice $v \in V(G)$, $i \in L_v$ y
 $check_{uv,0}(n) = (n \geq 1)$ para toda arista $uv \in E(G)$.

Notemos que CUBRIMIENTO POR ARISTAS (en inglés, *edge cover*) es esencialmente lo mismo pero intercambiando vértices y aristas.

Respecto al segundo tipo de problemas, como dos aristas adyacentes en $E(G)$ están a distancia 2 en $J(G)$, podemos usar las ideas que empleamos anteriormente para DOMINACIÓN SEMI-TOTAL: los vecinos en $J(G)$ de las aristas en $E(G)$ (que son vértices en $V(G)$) son los encargados de verificar los requerimientos de las aristas en $E(G)$. Lo ilustraremos para el problema de EMPAREJAMIENTO MÁXIMO (*maximum matching*):

- $L_v = \{0\}$ para todo vértice $v \in V(G)$ y
 $L_{uv} = \{0, 1\}$ para toda arista $uv \in E(G)$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{R}^{-\infty}, \geq, +)$,
- $w(\nu, c) = c(\nu)$ para todo elemento $\nu \in V(J(G))$,
- $check(v, c) = \left(\sum_{u \in N_G(v)} c(uv) \leq 1 \right)$ para todo vértice $v \in V(G)$ y
 $check(uv, c) = \text{TRUE}$ para toda arista $uv \in E(G)$,
- $\mathcal{N}_{\nu,i} = \{0, 1\}$ para todos $\nu \in V(J(G))$, $i \in L_\nu$,
- $n \boxplus^{\nu,i} n' = \min(n + n', 1)$ para todos $\nu \in V(J(G))$, $i \in L_\nu$,
- $newN_{\nu,i}(\nu', j) = j$ para todos $\nu, \nu' \in V(J(G))$, $i \in L_\nu$, $j \in L_{\nu'}$,
- $w_{\nu,i}(n) = i$ para todos $\nu \in V(J(G))$, $i \in L_\nu$,
- $check_{v,0}(n) = (n \leq 1)$ para todo vértice $v \in V(G)$ y
 $check_{uv,i}(n) = \text{TRUE}$ para toda arista $uv \in E(G)$, $i \in L_{uv}$.

Capítulo 8

Relación con otros marcos generales

8.1. Problemas LCVS y LCVP

Sean σ y ρ conjuntos finitos o co-finitos de enteros no negativos. Un subconjunto de vértices de un grafo G es un *conjunto* (σ, ρ) si para todo vértice $v \in S$ tenemos $|N(v) \cap S| \in \sigma$ y para todo vértice $v \in V(G) - S$ tenemos $|N(v) \cap S| \in \rho$. Los llamados *problemas localmente verificables de subconjuntos de vértices* (en inglés, *locally checkable vertex subset*, abreviado LCVS), definidos por Telle en [164], consisten en encontrar un conjunto (σ, ρ) mínimo o máximo en un grafo dado, con la posibilidad de que este grafo sea pesado.

Una generalización de estos problemas, definida también en [164], pregunta por una partición del conjunto de vértices en un número constante q de clases, cada una de las cuales debe satisfacer una cierta propiedad (σ, ρ) , como se describe a continuación. Una *matriz de restricciones de grado D* es una matriz de $q \times q$ cuyas entradas son conjuntos finitos o co-finitos de enteros no negativos. Una *D -partición* de un grafo G es una partición $\{V_1, V_2, \dots, V_q\}$ de $V(G)$ tal que, para todos i, j con $1 \leq i, j \leq q$ y para todo $v \in V_i$, vale que $|N(v) \cap V_j| \in D[i, j]$. Un *problema localmente verificable de partición de vértices* (en inglés, *locally checkable vertex partitioning*, abreviado LCVP) consiste en decidir si G tiene una D -partición. También existen variantes de optimización de este problema en grafos pesados. Los *problemas LCVP de distancia r* se definen en [121] como una generalización de los problemas LCVP en los cuales se considera, para cada vértice $v \in V(G)$, su vecindad de radio r , $N_G^r(v)$, en lugar de $N_G(v)$.

En [165], Telle y Proskurowski probaron que los problemas LCVP son FPT parametrizados por treewidth. Este resultado fue generalizado en [50], donde Bui-Xuan, Telle y Vatshelle presentaron un algoritmo basado en programación dinámica para resolver problemas LCVP dado un árbol binario de descomposición del grafo de entrada, cuya complejidad es FPT parametrizada por boolean-width. En [121], Jaffke, Kwon, Strømme y Telle extienden este resultado, mostrando que este mismo algoritmo se puede adaptar a los problemas LCVP de distancia r , y que entonces estos son XP parametrizados por la mim-width de un árbol binario de descomposición dado.

En [95], Gerber y Kobler estudiaron una variante de LCVP donde las entradas de la matriz D son conjuntos de enteros consecutivos (no necesariamente acotados por alguna constante) y cada vértice v tiene un conjunto asociado $\rho(v) \subseteq \llbracket 1, q \rrbracket$ tal que si $v \in V_i$ entonces $i \in \rho(v)$ (en otras palabras, cada vértice tiene asociada una lista de clases a las que puede pertenecer, tal como ocurre en coloreo por listas). Ellos probaron

que estos problemas son XP parametrizados por clique-width.

Los problemas LCVP pueden ser modelados como problemas 1-localmente verificables de la siguiente manera:

- $\text{COLORS} = \llbracket 1, q \rrbracket$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{+\infty}, \leq, +)$ en el caso de problemas de minimización, o $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$ en el caso de problemas de maximización,
- $w(v, c) = w_{v, c(v)}$ para algún valor $w_{v, c(v)}$ dado en la entrada o según el problema específico, y
- $\text{check}(v, c) = (\forall j \in \llbracket 1, q \rrbracket. |N_G(v) \cap c^{-1}(j)| \in D[c(v), j])$.

Con este modelo, las funciones de costo y verificación son contadoras de colores con las siguientes funciones asociadas:

$$w'(v, i, \mu) = w_{v, i},$$

$$\text{check}'(v, i, \mu) = (\forall j \in \llbracket 1, q \rrbracket. \mu_j \in D[i, j]).$$

Notemos además que, como los conjuntos $D[i, j]$ son finitos o co-finitos, para verificar si $\mu_j \in D[i, j]$ alcanza con conocer el valor de μ_j hasta uno más que el máximo valor de $D[i, j]$ si este conjunto es finito, o el máximo valor del complemento de $D[i, j]$ si este conjunto es co-finito. Por lo tanto, el modelo dado es d -estable, siendo d el máximo entre todos estos valores mencionados para los conjuntos $D[i, j]$. De esta manera, los Corolarios 3.1.8, 4.1.11 y 5.1.4 nos permiten obtener los mismos resultados de [50, 121, 165] respecto a la parametrización de los problemas LCVP según treewidth, clique-width y mim-width. Respecto a la variante considerada en [95], sólo podemos decir que es contadora de colores y entonces el Corolario 4.1.9 da los mismos resultados que los presentados en [95]. Este análisis también lo podemos hacer para los problemas LCVP de distancia r , de acuerdo a lo expuesto en las Secciones 3.3, 4.3 y 5.1.1.

Los problemas LCPV tienen la característica de que la propiedad local que todo vértice debe satisfacer considera independientemente la cantidad de vecinos en cada clase. Si bien muchos problemas localmente verificables de interés son expresables como problemas LCVP, existen otros igualmente relevantes que no admiten tal caracterización pero que sí pertenecen a nuestro marco (e incluso algunos son d -estables). Por ejemplo, el problema de $\{k\}$ -DOMINACIÓN (ver definición en el Apéndice A), ya que la propiedad local involucra la *suma* de todos los valores asignados a los vecinos. En el Capítulo 7 podemos encontrar aún más ejemplos de este tipo de problemas. Por lo tanto, los resultados de esta tesis efectivamente generalizan los obtenidos previamente respecto a problemas LCVP.

8.2. Problemas LCC y ECC

En [32, 33] Bodlaender definió los problemas llamados *de composición de condiciones locales* (en inglés, *local condition composition*, abreviado LCC) y *de composición de condiciones en aristas* (*edge condition composition*, ECC), y mostró que los primeros se pueden resolver en tiempo polinomial en grafos de treewidth y grado acotados, y que

los segundos se pueden resolver en tiempo polinomial en grafos de treewidth acotada. A continuación presentaremos las definiciones originales de estos problemas. Notamos $M_G^k(v)$ al conjunto de aristas cuyos extremos están a distancia a lo sumo k de v en G .

Definición 8.2.1 ([32, Definición 2.9]). Sea Π un problema de decisión en grafos, y sean D_Π el conjunto de instancias de Π , Y_Π el conjunto de instancias para las cuales la respuesta es “sí”, y $s : D_\Pi \rightarrow \mathbb{N}$ una función que asigna tamaños a instancias de Π . Decimos que Π es un *problema básico de condiciones locales*, si y sólo si existen

- enteros no negativos $m, c \in \mathbb{N}$,
- m monoides conmutativos $(M^1, \otimes^1), \dots, (M^m, \otimes^m)$, y
- una tupla $(M^{m+1}, \otimes^{m+1}, \preceq)$ tal que $(M^{m+1}, \otimes^{m+1})$ es un monoide conmutativo, $(M^{m+1}, \preceq)$ es un orden total y $x \preceq y \Rightarrow x \otimes^{m+1} z \preceq y \otimes^{m+1} z$ para todos $x, y, z \in M^{m+1}$

tales que

- cada $D \in D_\Pi$ es de la forma $(G, (X, Y, R_1, \dots, R_m, K, I))$, donde
 - G es un grafo no dirigido
 - X es un conjunto finito con $s(D) \geq |X|$
 - Y es un conjunto finito con $s(D) \geq |Y|$
 - para todo i , $1 \leq i \leq m$, R_i denota a un subconjunto de M^i
 - $K \in M^{m+1}$
- para todo i , $1 \leq i \leq m+1$, existe una función val_i , que asigna todas las 4-tuplas, que consisten en una instancia $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, un vértice $v \in V(G)$, y funciones $f : N_G^c[v] \rightarrow X$, $g : M_G^c(v) \rightarrow Y$, a elementos de M_i , tal que para toda (constante) $d \in \mathbb{N}^+$:
 1. existe un algoritmo que, en tiempo polinomial en $s(D)$, calcula $val_i(D, v, f, g)$ para todos $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $v \in V(G)$, $f : N_G^c[v] \rightarrow X$, $g : M_G^c(v) \rightarrow Y$ con $\Delta(G) \leq d$.
 2. si $1 \leq i \leq m$, existe un polinomio p_i tal que, para todos $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, con $\Delta(G) \leq d$ y subconjuntos $W \subseteq V(G)$, tenemos $|\{\bigotimes_{w \in W}^i val_i(D, w, f|_{N_G^c[w]}, g|_{M_G^c(w)}) \mid f : N_G^c[W] \rightarrow X, g : M_G^c(W) \rightarrow Y\}| \leq p_i(s(D))$.
 3. existe un algoritmo que, en tiempo polinomial en $s(D)$, calcula $a \otimes^i b$ para a, b tales que existen $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, con $\Delta(G) \leq d$, $W_1 \subseteq V(G)$, $W_2 \subseteq V(G)$, $W_1 \cap W_2 = \emptyset$, $f : N_G^c[W_1 \cup W_2] \rightarrow X$, $g : M_G^c(W_1 \cup W_2) \rightarrow Y$, y $a = \bigotimes_{w \in W_1}^i val_i(D, w, f|_{N_G^c[w]}, g|_{M_G^c(w)})$ y $b = \bigotimes_{w \in W_2}^i val_i(D, w, f|_{N_G^c[w]}, g|_{M_G^c(w)})$.
 4. si $i = m+1$ entonces existe un algoritmo que, en tiempo polinomial en $s(D)$, determina si $a \preceq b$ para a, b tales que existen $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, con $\Delta(G) \leq d$, $W \subseteq V(G)$, $f_1 : N_G^c[W] \rightarrow X$, $f_2 : N_G^c[W] \rightarrow X$, $g_1 : M_G^c(W) \rightarrow Y$, $g_2 : M_G^c(W) \rightarrow Y$, y $a = \bigotimes_{w \in W}^{m+1} val_{m+1}(D, w, f_1|_{N_G^c[w]}, g_1|_{M_G^c(w)})$ y $b = \bigotimes_{w \in W}^{m+1} val_{m+1}(D, w, f_2|_{N_G^c[w]}, g_2|_{M_G^c(w)})$ o $b = K$.

5. si $1 \leq i \leq m$, existe un algoritmo que, en tiempo polinomial en $s(D)$, determina si $a \in R_i$, para todos $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$ con $\Delta(G) \leq d$, $f: V(G) \rightarrow X$, $g: E(G) \rightarrow Y$ y $a = \bigotimes_{w \in V(G)}^i \text{val}_i(D, w, f|_{N_G^c[w]}, g|_{M_G^c(w)})$.
- Para todo $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $D \in Y_\Pi$ si y sólo si existen funciones $f: V(G) \rightarrow X$, $g: E(G) \rightarrow Y$, tales que
 1. $\forall i, 1 \leq i \leq m : \bigotimes_{v \in V(G)}^i \text{val}_i(D, v, f|_{N_G^c[v]}, g|_{M_G^c(v)}) \in R_i$
 2. $\bigotimes_{v \in V(G)}^{m+1} \text{val}_{m+1}(D, v, f|_{N_G^c[v]}, g|_{M_G^c(v)}) \leq K$.

Definición 8.2.2 ([32, Definición 2.10]). Sea Π un problema de decisión en grafos. Decimos que Π es un *problema de composición de condiciones locales* si y sólo si existen un problema básico de condiciones locales Π' y una transformación polinomial de Π a Π' . La clase de problemas de composición de condiciones locales se nota LCC.

Definición 8.2.3 ([32, Definición 2.11]). Sea Π un problema de decisión en grafos, y sean D_Π el conjunto de instancias de Π , Y_Π el conjunto de instancias para las cuales la respuesta es “sí”, y $s: D_\Pi \rightarrow \mathbb{N}$ una función que asigna tamaños a las instancias de Π . Decimos que Π es un *problema básico de condiciones en aristas*, si y sólo si existen

- un entero no negativo $m \in \mathbb{N}$,
- m monoides conmutativos $(M^1, \otimes^1), \dots, (M^m, \otimes^m)$, y
- una tupla $(M^{m+1}, \otimes^{m+1}, \preceq)$ tal que $(M^{m+1}, \otimes^{m+1})$ es un monoide conmutativo, $(M^{m+1}, \preceq)$ es un orden total y $x \preceq y \Rightarrow x \otimes^{m+1} z \preceq y \otimes^{m+1} z$ para todos $x, y, z \in M^{m+1}$

tales que

- cada $D \in D_\Pi$ es de la forma $(G, (X, Y, R_1, \dots, R_m, K, I))$, donde
 - G es un grafo no dirigido
 - X es un conjunto finito con $s(D) \geq |X|$
 - Y es un conjunto finito con $s(D) \geq |Y|$
 - para todo i , $1 \leq i \leq m$, R_i denota a un subconjunto de M^i
 - $K \in M^{m+1}$
- para todo i , $1 \leq i \leq m+1$, existe una función val_i , que asigna todas las 4-tuplas, que consisten en una instancia $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, una arista $uv \in E(G)$, y funciones $f: \{u, v\} \rightarrow X$, $g: \{uv\} \rightarrow Y$, a elementos de M_i , tal que:
 1. existe un algoritmo que, en tiempo polinomial en $s(D)$, calcula $\text{val}_i(D, e, f, g)$, para todos $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $uv \in E(G)$, $f: \{u, v\} \rightarrow X$, $g: \{uv\} \rightarrow Y$.
 2. si $1 \leq i \leq m$, existe un polinomio p_i tal que para todos $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$ y $E' \subseteq E(G)$, tenemos $|\{\bigotimes_{uv \in E'}^i \text{val}_i(D, uv, f(u), f(v), g(uv)) \mid f: N_G(E) \rightarrow X, g: E \rightarrow Y\}| \leq p_i(s(D))$.

3. existe un algoritmo que, en tiempo polinomial en $s(D)$, calcula $a \otimes^i b$ para a, b tales que existen $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $E_1 \subseteq E(G)$, $E_2 \subseteq E(G)$, $E_1 \cap E_2 = \emptyset$, $f: N_G(E_1 \cup E_2) \rightarrow X$, $g: E_1 \cup E_2 \rightarrow Y$, y $a = \bigotimes_{uv \in E_1}^i \text{val}_i(D, uv, f(u), f(v), g(uv))$ y $b = \bigotimes_{uv \in E_2}^i \text{val}_i(D, uv, f(u), f(v), g(uv))$.
 4. si $i = m + 1$ entonces existe un algoritmo que, en tiempo polinomial en $s(D)$, determina si $a \preceq b$ para a, b tales que existen $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $E' \subseteq E(G)$, $f_1: N_G(E') \rightarrow X$, $f_2: N_G(E') \rightarrow X$, $g_1: E' \rightarrow Y$, $g_2: E' \rightarrow Y$, y $a = \bigotimes_{uv \in E'}^{m+1} \text{val}_{m+1}(D, uv, f_1(u), f_1(v), g_1(uv))$ y $b = \bigotimes_{uv \in E'}^{m+1} \text{val}_{m+1}(D, uv, f_2(u), f_2(v), g_2(uv))$ o $b = K$.
 5. si $1 \leq i \leq m$, existe un algoritmo que, en tiempo polinomial en $s(D)$, determina si $a \in R_i$, para todos $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $f: V(G) \rightarrow X$, $g: E(G) \rightarrow Y$ y $a = \bigotimes_{uv \in E(G)}^i \text{val}_i(D, uv, f(u), f(v), g(uv))$.
- Para todo $D = (G, (X, Y, R_1, \dots, R_m, K, I)) \in D_\Pi$, $D \in Y_\Pi$ si y sólo si existen funciones $f: V(G) \rightarrow X$, $g: E(G) \rightarrow Y$, tales que
 1. $\forall i, 1 \leq i \leq m: \bigotimes_{uv \in E(G)}^i \text{val}_i(D, uv, f(u), f(v), g(uv)) \in R_i$
 2. $\bigotimes_{uv \in E(G)}^{m+1} \text{val}_{m+1}(D, uv, f(u), f(v), g(uv)) \leq K$.

Definición 8.2.4 ([32, Definición 2.12]). Sea Π un problema de decisión en grafos. Decimos que Π es un *problema de composición de condiciones en aristas*, si y sólo si existen un problema básico de condiciones en aristas Π' y una transformación polinomial de Π a Π' . La clase de problemas de composición de condiciones en aristas se nota ECC.

Teorema 8.2.5 ([32, Teorema 2.5]). $ECC \subseteq LCC$.

Teorema 8.2.6 ([32, Teorema 3.7]). (i) Sea $\Pi \in LCC$, y sean $k, d \in \mathbb{N}^+$. Sea Θ una clase de grafos tal que $G \in \Theta \Rightarrow \Delta(G) \leq d \wedge \text{tw}(G) \leq k$. Luego, Π restringido a Θ puede resolverse en tiempo polinomial.

(ii) Sea $\Pi \in ECC$, y sea $k \in \mathbb{N}^+$. Sea Θ una clase de grafos tal que $G \in \Theta \Rightarrow \text{tw}(G) \leq k$. Luego, Π restringido a Θ puede resolverse en tiempo polinomial.

Consideremos un problema $\Pi \in ECC$. Por definición, existe un problema básico de condiciones en aristas Π' tal que Π puede ser reducido polinomialmente a Π' . Vamos a mostrar que Π' puede ser reducido polinomialmente a un problema 1-localmente verificable con propiedades globales de composición de condiciones locales en el grafo dentado $J(G)$ del grafo G de entrada de Π' . Sea $(M^*, \otimes^*, \preceq^*)$ obtenido al agregar un elemento máximo a $(M^{m+1}, \otimes^{m+1}, \preceq)$. Luego,

- $L_v = X$ para todo vértice $v \in V(G)$, y
 $L_{uv} = L_u \times L_v \times Y$ para toda arista $uv \in E(G)$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (M^*, \preceq^*, \otimes^*)$,
- $W_{v,x} = I_\oplus$ para todo vértice $v \in V(G)$, y
 $W_{uv,(x_u,x_v,y)} = \text{val}_{m+1}(D, uv, x_u, x_v, y)$ para toda arista $uv \in E(G)$,

- $check(v, c) = \text{TRUE}$ para todo vértice $v \in V(G)$ y toda asignación de colores c válida en $N_{J(G)}[v]$, y
 $check(uv, c) = (c(uv)_1 = c(u) \wedge c(uv)_2 = c(v))$ para toda arista $uv \in E(G)$ y toda asignación de colores c válida en $N_{J(G)}[e]$, y
- propiedades globales de composición de condiciones locales (sobre un coloreo c):
 $\bigotimes_{uv \in E(G)}^i val_i(D, uv, c(uv)_1, c(uv)_2, c(uv)_3) \in R_i$, para todo $i \in \llbracket 1, m \rrbracket$.

Con un enfoque similar podemos reducir polinomialmente un problema $\Pi \in LCC$, restringido a grafos de treewidth y grados acotados, a un problema r -localmente verificable con propiedades globales de composición de condiciones locales en el grafo dentado $J(G)$ del grafo G de entrada de Π (para algún r apropiado). Sea $(M^*, \otimes^*, \preceq^*)$ una tupla obtenida al agregar un elemento máximo a $(M^{m+1}, \otimes^{m+1}, \preceq)$. Para todo $v \in V(G)$, sean $\mathbb{F}_v$ y $\mathbb{G}_v$, respectivamente, los conjuntos de todas las funciones $f: N_G^r[v] \rightarrow X$ y $g: M_G^r(v) \rightarrow Y$ (notemos que $\mathbb{F}_v$ y $\mathbb{G}_v$ son conjuntos finitos que se pueden calcular en tiempo polinomial). Luego,

- $L_v = \mathbb{F}_v \times \mathbb{G}_v$ para todo vértice $v \in V(G)$, y
 $L_{uv} = Y$ para toda arista $uv \in E(G)$,
- $(\text{WEIGHTS}, \preceq, \oplus) = (M^*, \preceq^*, \otimes^*)$,
- $W_{v,(f,g)} = val_{m+1}(D, v, f, g)$ para todo vértice $v \in V(G)$, y
 $W_{uv,y} = I_{\oplus}$ para toda arista $uv \in E(G)$;
- para todo vértice $v \in V(J(G))$ y toda asignación de colores c válida en $N_{J(G)}^r[v]$, $check(v, c)$ verifica que $f_u(x) = f_w(x)$ y $g_u(y) = g_w(y)$ (donde $(f_u, g_u) = c(u)$ y $(f_w, g_w) = c(w)$) para todos $u, w \in N_{J(G)}^r[v] \cap V(G)$, $x \in N_{J(G)}^r[v] \cap N_G^r[u] \cap N_G^r[w]$ y $y \in N_{J(G)}^r[v] \cap M_G^r(u) \cap M_G^r(w)$, y
- propiedades globales de composición de condiciones locales (sobre un coloreo c):
 $\bigotimes_{v \in V(G)}^i val_i(D, v, c(v)_1, c(v)_2) \in R_i$, para todo $i \in \llbracket 1, m \rrbracket$.

Notemos que en ambos casos podemos dar, de manera directa, sistemas de vecindades parciales polinomiales. Por lo tanto, estas reducciones junto con los resultados del Capítulo 3 nos permiten obtener los resultados de Bodlaender en [32, 33]. Si bien no se menciona explícitamente en estos artículos, los resultados generales refieren a algoritmos XP parametrizados por treewidth (en el caso de problemas ECC) y XP parametrizados por treewidth más grado (problemas LCC), y si tanto $|X|$ como $|Y|$ están acotados por alguna constante entonces los algoritmos son FPT. Los resultados de esta tesis nos permiten decir precisamente esto mismo.

Es más, nuestros resultados son *equivalentes* respecto a la parametrización por treewidth más grado, ya que podemos pensar a la función $check$ como un caso particular de las funciones val_i y a w como val_{m+1} . Sin embargo, dado que aún en el caso $c = 1$ la complejidad del algoritmo de [32] sigue dependiendo exponencialmente del grado, los resultados del Capítulo 3 son efectivamente más generales. Por otro lado, también analizamos la complejidad para más parámetros de anchura (clique-width y mim-width). Incluso podemos responder automáticamente algunas de las preguntas abiertas planteadas en [33, Tabla 1]. Por ejemplo, Bodlaender mostró que el problema *máximo subgrafo conexo de grado acotado* (*maximum degree-bounded connected subgraph*, en

inglés) es LCC y por lo tanto resoluble en tiempo polinomial en grafos de treewidth y grado acotados, pero dejó abierta la complejidad para grafos de treewidth acotada. En este problema, dado un grafo G y un entero positivo d , se busca un subgrafo conexo de grado máximo a lo sumo d , con conjunto de vértices igual a $V(G)$ y con la mayor cantidad posible de aristas. Notemos que se puede modelar fácilmente como un problema 1-localmente verificable en el grafo $S(G)$: los conjuntos de colores disponibles son $L_v = \{1\}$ para todo $v \in V(G)$ y $L_e = \{0, 1\}$ para toda $e \in E(G)$, los costos son $w(v, c) = 1$ si $v \in E(G)$ y $c(v) = 1$, y $w(v, c) = 0$ en cualquier otro caso, la función de verificación está dada por $check(v, c) = \left(\sum_{e \in N_{S(G)}(v)} c(e) \leq d \right)$ para todo $v \in V(G)$ y $check(e, c) = \text{TRUE}$ para toda $e \in E(G)$, y se incluye una propiedad global para pedir que la clase de color 1 sea conexa. Por los resultados del Capítulo 3, este problema es resoluble en tiempo polinomial en grafos de treewidth acotada.

8.3. Lógica A&C DN

La *lógica de vecindad a distancia con restricciones de aciclicidad y conectividad* (en inglés, *distance neighborhood logic with acyclicity and connectivity constraints*, *A&C DN*) fue presentada por Bergougnoux, Dreier y Jaffke en [26], donde además prueban que todo problema expresable mediante una fórmula de longitud constante en dicha lógica es XP parametrizado por mim-width. Esta lógica está basada en la famosa lógica monádica de segundo orden MSO_1 del llamado Teorema de Courcelle para clique-width [67].

Los grafos se pueden considerar como estructuras lógicas y así podemos expresar formalmente sus propiedades mediante fórmulas lógicas. Una fórmula en lógica MSO_1 es una fórmula en lógica de segundo orden sobre grafos donde las cuantificaciones ($\forall$ y $\exists$) sólo pueden ser sobre vértices o conjuntos de vértices. La *lógica MSO_1 existencial* es la restricción de MSO_1 donde no se permiten cuantificadores universales ni negaciones de términos que contengan cuantificadores existenciales.

A continuación transcribiremos la definición formal de la sintaxis de la lógica A&C DN dada en [26]. Los *términos de vecindad* se definen de la siguiente manera:

- toda variable de conjunto X es un término de vecindad,
- para todo término de vecindad t y enteros positivos d y r , $N_d^r(t)$ es un término de vecindad,
- toda relación unaria P es un término de vecindad,
- $\emptyset$ es un término de vecindad, y
- si t y t' son términos de vecindad entonces $\bar{t}$, $t \cap t'$, $t \cup t'$ y $t \setminus t'$ son términos de vecindad.

La lógica A&C DN se define entonces como la extensión de lógica MSO_1 existencial con las siguientes reglas:

- si t es un término de vecindad y m un entero no negativo, entonces $|t| = m$, $|t| \leq m$ y $|t| \geq m$ son fórmulas de esta lógica,

- si t y t' son términos de vecindad entonces $t = t'$, $t \subseteq t'$ y $t \supseteq t'$ son fórmulas de esta lógica, y
- si t es un término de vecindad, entonces $\text{acy}(t)$ y $\text{con}(t)$ son fórmulas de esta lógica.

Una relación unaria P representa un conjunto fijo de vértices, y $\emptyset$ el conjunto vacío. Sea t un término que representa un conjunto S de vértices de un grafo. Para enteros positivos d y r , el término $N_d^r(t)$ denota al conjunto de todos los vértices v para el cual existen al menos d vértices en S cuya distancia a v es al menos 1 y a lo sumo r . Además, $\text{acy}(t)$ y $\text{con}(t)$ expresan que el conjunto S induce un subgrafo acíclico y conexo, respectivamente. Las fórmulas $|t| = m$, $|t| \leq m$ y $|t| \geq m$ expresan la comparación del tamaño de S con m . Sea t' un término que representa un conjunto S' , luego $t = t'$, $t \subseteq t'$ y $t \supseteq t'$ expresan las respectivas comparaciones de los conjuntos S y S' . El resto de la semántica se hereda de la de MSO_1 .

Por ejemplo, la fórmula $\exists X. P \subseteq X \cup N_1^1(X)$ expresa la existencia de un conjunto X que domina a los vértices en el conjunto dado por P . La fórmula $\exists X. |X| \geq 5 \wedge N_2^1(X) \cap X = \emptyset$ expresa la existencia de un conjunto X de al menos 5 vértices tal que cada vértice en X tiene a lo sumo un vecino en X .

Para una fórmula φ en lógica A&C DN, $|\varphi|$ denota la longitud de φ , $d(\varphi)$ denota al mayor entero d tal que existe un término $N_d^1(\cdot)$ en φ , $R(\varphi)$ denota al conjunto de enteros r tales que existe un término $N^r(\cdot)$ en φ y $r(\varphi)$ denota al máximo elemento de $R(\varphi)$. El principal resultado de [26] indica que todo problema de optimización en grafos expresable mediante una fórmula φ en lógica A&C DN puede resolverse en tiempo $n^{O(w \cdot d(\varphi) \cdot |\varphi|^2)}$, más aún $n^{O(w \cdot d(\varphi) \cdot |\varphi|)}$ si $r(\varphi) = O(1)$, donde n es la cantidad de vértices del grafo de entrada y w es la mim-width de un árbol binario de descomposición asociado al grafo de entrada.

En [26] también demuestran que toda fórmula A&C DN φ puede ser expresada como una fórmula A&C CDN φ' , la cual es una fórmula de la forma $\exists X_1 \dots X_k. \psi$ donde ψ es una combinación booleana (es decir, una expresión que usa $\wedge$, $\vee$ y $\neg$) de los siguientes tipos de fórmulas primitivas:

1. $(P = X_i)$, para un P fijo y alguna variable X_i ,
2. $(X_i = X_j)$, para algunas variables X_i, X_j ,
3. $(X_i = \overline{X_j})$, para algunas variables X_i, X_j ,
4. $(X_i \cap X_j = X_h)$, para algunas variables X_i, X_j, X_h ,
5. $(N_d^r(X_i) = X_j)$, para algunas variables X_i, X_j y enteros positivos d y r ,
6. $(|X_i| \leq m)$, para alguna variable X_i y un entero positivo m ,
7. $\text{acy}(X_i)$, para alguna variable X_i ,
8. $\text{con}(X_i)$, para alguna variable X_i .

Más aún, la longitud de φ' es a lo sumo 10 veces la longitud de φ , $d(\varphi') = d(\varphi)$ y $R(\varphi') = R(\varphi)$ ([26, Observación 3.1]). Esta reformulación es útil para desarrollar

algoritmos sobre esta lógica y en particular la utilizaremos para comparar esta lógica con nuestro marco.

A continuación mostraremos que el conjunto de problemas que se pueden modelar utilizando fórmulas A&C DN de longitud constante y $d(\cdot)$ constante es equivalente al conjunto de problemas que se pueden modelar utilizando problemas localmente verificables con cantidad constante de colores, funciones de verificación y peso que son d -estables a t distancias (para algunas constantes d y t), y propiedades globales de tamaño, conectividad y aciclicidad.

Comenzaremos mostrando cómo modelar un problema expresable mediante una fórmula en lógica A&C DN pero esta vez utilizando problemas localmente verificables con propiedades globales de tamaño, conectividad y aciclicidad.

Lema 8.3.1 ([103, Lema 4.1]). *Sea Π un problema expresable mediante una fórmula φ en lógica A&C DN. Luego, existen enteros positivos $p \leq 2^{10|\varphi|}$, $q \leq 2^{10|\varphi|}$ y $t \leq |\varphi|$, y un conjunto $\{\Pi_1, \dots, \Pi_p\}$ de problemas localmente verificables con q colores, funciones de peso y verificación $d(\varphi)$ -estables a t distancias, y propiedades globales de peso, conectividad y aciclicidad, tales que, para una entrada dada, el óptimo entre todas las soluciones óptimas de $\Pi_1, \dots, \Pi_p$ es equivalente a la solución óptima de Π .*

Demostración. Sean G el grafo de entrada y ϕ una fórmula en lógica A&C CDN equivalente a φ , donde $\phi = (\exists X_1 \dots X_k. \psi)$ y ψ es una combinación booleana de fórmulas primitivas de los ocho tipos vistos más arriba. Sea $A = A_1 \cup \dots \cup A_8$ el conjunto de todas las fórmulas primitivas en ψ , donde A_i contiene exactamente las fórmulas primitivas que son del tipo i , para cada $i \in [1, 8]$.

Hay a lo sumo $|\psi|$ fórmulas primitivas en A y por lo tanto a lo sumo $2^{|\psi|}$ asignaciones de valores de verdad $T: A \rightarrow \text{BOOL}$. Para cada tal asignación que satisface ψ , vamos a mostrar cómo modelar funciones de verificación y peso, y propiedades globales de tamaño, conectividad y aciclicidad. Sea $T: A \rightarrow \text{BOOL}$ una asignación de valores de verdad que satisface ψ . Para cada $i \in [1, 8]$, sea A_i^+ el conjunto de todas las fórmulas primitivas de A_i a las cuales T les asigna TRUE. También notemos $A_i^- = A_i - A_i^+$. En otras palabras, el conjunto A_i^+ contiene todas las fórmulas primitivas de A_i que tienen que valer para satisfacer ψ según lo indicado por T . Análogamente, A_i^- contiene aquellas fórmulas primitivas de A_i que no deben valer, según T .

Definimos R como $R(\varphi)$ si este conjunto no es vacío, y en caso contrario definimos R como $\{1\}$. Notemos que el tamaño de R no es mayor a la longitud de φ .

Definimos $\text{COLORS} = \mathcal{P}([1, k]) \times \mathcal{P}(A_1) \times \mathcal{P}(A_5)$. Notemos que $|\text{COLORS}| = 2^k \cdot 2^{|A_1|} \cdot 2^{|A_5|} \leq 2^{|\phi|} \leq 2^{10|\varphi|}$. Intuitivamente, si un vértice v recibe el color (S, F_1, F_5) , entonces S representa el subconjunto de las variables $X_1, \dots, X_k$ en las cuales v está contenido, y F_1 y F_5 indican cuáles de las fórmulas en A_1 y A_5 , respectivamente, son localmente satisfechas en v . Vamos a definir una función *check* que sólo verifica si F_1 y F_5 cumplen lo mencionado anteriormente. Esta función será $d(\varphi)$ -estable a $|R|$ distancias $r_1, \dots, r_{|R|} \in R$. Para cada $v \in V(G)$, $(S, F_1, F_5) \in \text{COLORS}$ y $\bar{\mu} \in \mathbb{N}^{|\text{COLORS}| \times |R|}$ (cuyas entradas notamos mediante $\bar{\mu}_{(S', F'_1, F'_5)}$ para cualquier $r \in R$ y $(S', F'_1, F'_5) \in \text{COLORS}$), definimos $\text{check}(v, (S, F_1, F_5), \bar{\mu}) = \text{TRUE}$ si y sólo si se cumplen las siguientes dos condiciones:

- para todo $(P = X_i) \in A_1$, tenemos que $(P = X_i) \in F_1$ si y sólo si $(i \in S)$ y $(v \in P)$ son o bien ambos verdaderos o bien ambos falsos,

- para todo $(N_d^r(X_i) = X_j) \in A_5$, tenemos que $(N_d^r(X_i) = X_j) \in F_5$ si y sólo si $(j \in S)$ y $\left(\sum_{F'_1 \subseteq A_1} \sum_{F'_5 \subseteq A_5} \sum_{S' \subseteq [1, k], i \in S'} \bar{\mu}_{S', F'_1, F'_5}^r \geq d\right)$ son o bien ambos verdaderos o bien ambos falsos.

Definimos las siguientes propiedades globales de tamaño, conectividad y aciclicidad, las cuales van a terminar de garantizar que se satisface ψ .

- Para cada $(P = X_i) \in A_1^+$ y cada $(S, F_1, F_5) \in \text{COLORS}$ donde F_1 no contiene a $(P = X_i)$, el tamaño de la clase de color (S, F_1, F_5) es 0.
- Para cada $(P = X_i) \in A_1^-$, existe un color $(S, F_1, F_5) \in \text{COLORS}$ tal que F_1 no contiene a $(P = X_i)$ y el tamaño de la clase de color (S, F_1, F_5) es al menos 1.
- Para cada $(X_i = X_j) \in A_2^+$ y cada $(S, F_1, F_5) \in \text{COLORS}$ donde $|S \cap \{i, j\}| = 1$, el tamaño de la clase de color (S, F_1, F_5) es 0.
- Para cada $(X_i = X_j) \in A_2^-$, existe un color $(S, F_1, F_5) \in \text{COLORS}$ tal que $|S \cap \{i, j\}| = 1$ y el tamaño de la clase de color (S, F_1, F_5) es al menos 1.
- Para cada $(X_i = \overline{X_j}) \in A_3^+$ y cada $(S, F_1, F_5) \in \text{COLORS}$ donde $|S \cap \{i, j\}| \neq 1$, el tamaño de la clase de color (S, F_1, F_5) es 0.
- Para cada $(X_i = \overline{X_j}) \in A_3^-$, existe un color $(S, F_1, F_5) \in \text{COLORS}$ tal que $|S \cap \{i, j\}| \neq 1$ y el tamaño de la clase de color (S, F_1, F_5) es al menos 1.
- Para cada $(X_i \cap X_j = X_h) \in A_4^+$ y cada $(S, F_1, F_5) \in \text{COLORS}$ donde $S \cap \{i, j, h\} \in \{\{i, j\}, \{i, h\}, \{j, h\}, \{h\}\}$, el tamaño de la clase de color (S, F_1, F_5) es 0.
- Para cada $(X_i \cap X_j = X_h) \in A_4^-$, existe un color $(S, F_1, F_5) \in \text{COLORS}$ tal que $S \cap \{i, j, h\} \in \{\{i, j\}, \{i, h\}, \{j, h\}, \{h\}\}$ y el tamaño de la clase de color (S, F_1, F_5) es al menos 1.
- Para cada $(N_d^r(X_i) = X_j) \in A_5^+$ y cada $(S, F_1, F_5) \in \text{COLORS}$ donde F_5 no contiene a $(N_d^r(X_i) = X_j)$, el tamaño de la clase de color (S, F_1, F_5) es 0.
- Para cada $(N_d^r(X_i) = X_j) \in A_5^-$, existe un color $(S, F_1, F_5) \in \text{COLORS}$ tal que F_5 no contiene a $(N_d^r(X_i) = X_j)$ y el tamaño de la clase de color (S, F_1, F_5) es al menos 1.
- Para cada $(|X_i| \leq m) \in A_6^+$, la suma de los tamaños de todas las clases de colores (S, F_1, F_5) con $i \in S$ es a lo sumo m .
- Para cada $(|X_i| \leq m) \in A_6^-$, la suma de los tamaños de todas las clases de colores (S, F_1, F_5) con $i \in S$ es al menos $m + 1$.
- Para cada $\text{acy}(X_i) \in A_7^+$, el conjunto de todos los vértices de colores (S, F_1, F_5) , donde $i \in S$, induce un subgrafo acíclico.
- Para cada $\text{acy}(X_i) \in A_7^-$, el conjunto de todos los vértices de colores (S, F_1, F_5) , donde $i \in S$, induce un subgrafo no-acíclico.
- Para cada $\text{con}(X_i) \in A_8^+$, el conjunto de todos los vértices de colores (S, F_1, F_5) , donde $i \in S$, induce un subgrafo conexo.

- Para cada $\text{con}(X_i) \in A_g^-$, el conjunto de todos los vértices de colores (S, F_1, F_5) , donde $i \in S$, induce un subgrafo no-conexo.

En [26] se considera que el grafo de entrada G tiene un peso asociado $w(v, S) \in \mathbb{N}$ para cada vértice $v \in V(G)$ y cada conjunto $S \subseteq \llbracket 1, k \rrbracket$, y el peso de una k -tupla $B \in \mathcal{P}(V(G))^k$ se define como $\sum_{v \in V(G)} w(v, \{i : v \in B_i\})$. Luego, podemos considerar el conjunto de pesos $(\text{WEIGHTS}, \preceq, \oplus) = (\mathbb{N}^{-\infty}, \geq, +)$ y definir la función de peso como $w(v, (S, F_1, F_5), \bar{\mu}) = w(v, S)$ para cada $v \in V(G)$, $(S, F_1, F_5) \in \text{COLORS}$ y $\bar{\mu} \in \mathbb{N}^{|\text{COLORS}| \times |R|}$.

Para obtener una solución óptima del problema Π , tenemos que calcular la solución óptima del problema localmente verificable asociado a cada asignación T de valores de verdad a las fórmulas primitivas que satisface ψ , y luego simplemente elegir la mejor solución obtenida entre todas. Como mencionamos arriba, la cantidad de asignaciones T posibles es a lo sumo $2^{|\psi|}$ y, por [26, Observación 3.1], esto es a lo sumo $2^{10|\varphi|}$. $\square$

Ahora probaremos que todo problema localmente verificable con q colores, funciones de peso y verificación que son d -estables a t distancias, y propiedades globales de tamaño, conectividad y aciclicidad, puede ser expresado mediante fórmulas en lógica A&C DN cuyas longitudes están acotadas por una función en d , q y t únicamente. Este resultado, junto con el Teorema 1.2 de [26], implica que tal problema es XP parametrizado por la mim-width de un árbol binario de descomposición dado, cuando tanto d , q como t están acotados por una constante. Más aún, nos permite combinar estos problemas con cualquier otra propiedad potencialmente expresable en alguna extensión de esta lógica. Sin embargo, cuando el problema no tiene propiedades globales de aciclicidad entonces la complejidad de utilizar este método resulta un tanto peor que aplicar directamente el algoritmo del Capítulo 5, el cual, por otro lado, sólo puede manejar propiedades globales de tamaño y conectividad pero no de aciclicidad. En efecto, para un problema d -estable con q colores y ninguna propiedad global, la complejidad dada por el algoritmo del Capítulo 5 es $n^{O(wqd)}$ mientras que la complejidad que resulta de aplicar esta idea es $n^{O(wq^2d^2(d+1)^{2q})}$, siendo w la mim-width del árbol binario de descomposición dado. No obstante, ambos resultados implican que los problemas son XP parametrizados por w .

Lema 8.3.2 ([103, Lema 4.2]). *Consideremos un problema localmente verificable Π en un grafo G , con un conjunto de q colores $\text{COLORS} = \{a_1, \dots, a_q\}$, funciones de peso y verificación que son d -estables a t distancias $r_1, \dots, r_t$, propiedades globales de tamaño dadas por un conjunto $\mathcal{K} \subseteq \text{part}_q(|V(G)|)$, propiedades globales de conectividad positiva y negativa dadas por los conjuntos $\mathcal{C}^+, \mathcal{C}^- \subseteq \mathcal{P}(\llbracket 1, q \rrbracket)$, y propiedades globales de aciclicidad positiva y negativa dadas por $\mathcal{A}^+, \mathcal{A}^- \subseteq \mathcal{P}(\llbracket 1, q \rrbracket)$. Luego, existe un conjunto $\Phi = \{\varphi_1, \dots, \varphi_{|\mathcal{K}|}\}$ de fórmulas en lógica A&C DN tal que resolver Π es equivalente a encontrar el óptimo entre todas las soluciones óptimas de los problemas asociados a las fórmulas en Φ , y la longitud de cada una de estas fórmulas es $O(tq^2d(d+1)^{2qt})$.*

Demostración. Sea $K \in \mathcal{K}$. Vamos a mostrar cómo construir una fórmula φ_K en lógica A&C DN que expresa el problema de determinar el mínimo peso de un coloreo propio que satisface la propiedad global de tamaño dada por K y las propiedades globales de conectividad y aciclicidad dadas por $\mathcal{C}^+$, $\mathcal{C}^-$, $\mathcal{A}^+$ y $\mathcal{A}^-$. El conjunto Φ que buscamos es entonces $\{\varphi_K : K \in \mathcal{K}\}$. En efecto, una solución óptima de Π tiene que satisfacer alguna de las restricciones de tamaño en $\mathcal{K}$. La solución óptima de Π entonces tiene

que ser la mejor entre todas las soluciones óptimas a los problemas asociados a las fórmulas en Φ .

Sea $\mathcal{T} = \llbracket 1, q \rrbracket \times \llbracket 0, d \rrbracket^{qt}$. Para cada tupla $(j, \bar{\mu}) \in \mathcal{T}$, donde $j \in \llbracket 1, q \rrbracket$ y $\bar{\mu} = ((\bar{\mu}_1^1, \dots, \bar{\mu}_q^1), \dots, (\bar{\mu}_1^t, \dots, \bar{\mu}_q^t)) \in \llbracket 0, d \rrbracket^{qt}$, definimos el conjunto $P_{(j, \bar{\mu})} = \{v \in V(G) : \text{check}(v, a_j, \bar{\mu}) = \text{TRUE}\}$.

Primero, notemos que encontrar un coloreo propio de peso mínimo es equivalente a encontrar el menor peso de una partición de $V(G)$ en conjuntos X_T , con $T \in \mathcal{T}$, tales que se satisfacen las siguientes condiciones:

1. $\text{check}(v, a_j, \bar{\mu}) = \text{TRUE}$ (o, equivalentemente, $v \in P_{(j, \bar{\mu})}$) para todas las tuplas $(j, \bar{\mu}) \in \mathcal{T}$ y todos los vértices $v \in X_{(j, \bar{\mu})}$,
2. $\bar{\mu}_j^i = \min \left(d, \left| N_G^{r_i}(v) \cap \left(\bigcup_{\bar{\mu}' \in \llbracket 0, d \rrbracket^{qt}} X_{(j, \bar{\mu}')} \right) \right| \right)$ para todo $i \in \llbracket 1, t \rrbracket$ y todo $j \in \llbracket 1, q \rrbracket$, para todas las tuplas $(j, \bar{\mu}) \in \mathcal{T}$ y todos los vértices $v \in X_{(j, \bar{\mu})}$,
3. para todo $j \in \llbracket 1, q \rrbracket$, la cantidad de vértices en el conjunto $\bigcup_{\bar{\mu} \in \llbracket 0, d \rrbracket^{qt}} X_{(j, \bar{\mu})}$ es igual a K_j , es decir, la propiedad global de tamaño se satisface para toda clase de color,
4. para todo $C \in \mathcal{C}^+$ (respectivamente $\mathcal{C}^-$), el subgrafo inducido por el conjunto $\bigcup_{j \in C} \bigcup_{\bar{\mu} \in \llbracket 0, d \rrbracket^{qt}} X_{(j, \bar{\mu})}$ es conexo (respectivamente no conexo), es decir, se satisfacen las propiedades globales de conectividad,
5. para todo $A \in \mathcal{A}^+$ (respectivamente $\mathcal{A}^-$), el subgrafo inducido por el conjunto $\bigcup_{j \in A} \bigcup_{\bar{\mu} \in \llbracket 0, d \rrbracket^{qt}} X_{(j, \bar{\mu})}$ es acíclico (respectivamente no acíclico), es decir, se satisfacen las propiedades globales de aciclicidad,

donde los pesos son definidos como $w_{(j, \bar{\mu})}(v) = w(v, a_j, \bar{\mu})$. En otras palabras, partimos el conjunto de vértices de acuerdo al color que reciben y, para cada $j \in \llbracket 1, q \rrbracket$ y $r \in R$, el número (hasta d) de vértices de color a_j que tienen a distancia a lo sumo r .

En lo que sigue utilizaremos la siguiente fórmula definida en [26, Proposición 6.4]:

$$\text{part}(X_{T \in \mathcal{T}}) \equiv \overline{\bigcup_{T \in \mathcal{T}} X_T} = \emptyset \wedge \bigwedge_{T, T' \in \mathcal{T}} X_T \cap X_{T'} = \emptyset.$$

En otras palabras, $\text{part}(X_{T \in \mathcal{T}})$ evalúa a TRUE si y sólo si los conjuntos de vértices $X_{T \in \mathcal{T}}$ conforman una partición del conjunto $V(G)$. También utilizaremos la siguiente función, la cual se define para un entero positivo r y un conjunto finito o co-finito $\mu \subseteq \mathbb{N}$:

$$t_{r, \mu}(X) = \{v \in V(G) : |N_G^r(v) \cap X| \in \mu\}.$$

Esta función fue presentada en [26, Lema 6.1], donde se prueba que $t_{r, \mu}(X)$ puede ser expresada como una fórmula en lógica A&C DN de longitud $O(d(\mu)x)$, donde x es la longitud del término que representa X .

Consideremos la siguiente fórmula φ_K en lógica A&C DN formula. Afirmamos que expresa la existencia de una partición que satisface las 5 condiciones mencionadas arriba.

$$\begin{aligned} \varphi_K(X_{T \in \mathcal{T}}) \equiv & \text{part}(X_{T \in \mathcal{T}}) \wedge \text{proper}(X_{T \in \mathcal{T}}) \wedge \text{neigh}(X_{T \in \mathcal{T}}) \\ & \wedge \text{size}(X_{T \in \mathcal{T}}) \wedge \text{conn}(X_{T \in \mathcal{T}}) \wedge \text{acyc}(X_{T \in \mathcal{T}}). \end{aligned}$$

donde:

- $\text{proper}(X_{T \in \mathcal{T}}) \equiv \bigwedge_{T \in \mathcal{T}} X_T \subseteq P_T,$
- $\text{neigh}(X_{T \in \mathcal{T}}) \equiv \bigwedge_{(j, \bar{\mu}) \in \mathcal{T}} X_{(j, \bar{\mu})} \subseteq \left(\bigcap_{i \in [1, t]} \bigcap_{h \in [1, q]} t_{r_i, f(\bar{\mu}_h^i)} \left(\bigcup_{\bar{\mu}' \in [0, d]^{qt}} X_{(h, \bar{\mu}')} \right) \right),$
donde la función f se define como $f(x) = \{x\}$ si $x < d$ y $f(x) = \{y \in \mathbb{N} : y \geq x\}$ si no,
- $\text{size}(X_{T \in \mathcal{T}}) \equiv \left(\bigwedge_{j \in [q]} \left| \bigcup_{\bar{\mu} \in [0, d]^{qt}} X_{(j, \bar{\mu})} \right| = K_j \right),$
- $\text{conn}(X_{T \in \mathcal{T}}) \equiv \left(\bigwedge_{C \in \mathcal{C}^+} \text{con} \left(\bigcup_{j \in C} \bigcup_{\bar{\mu} \in [0, d]^{qt}} X_{(j, \bar{\mu})} \right) \right) \wedge \left(\bigwedge_{C \in \mathcal{C}^-} \neg \text{con} \left(\bigcup_{j \in C} \bigcup_{\bar{\mu} \in [0, d]^{qt}} X_{(j, \bar{\mu})} \right) \right),$
- $\text{acyc}(X_{T \in \mathcal{T}}) \equiv \left(\bigwedge_{A \in \mathcal{A}^+} \text{acy} \left(\bigcup_{j \in A} \bigcup_{\bar{\mu} \in [0, d]^{qt}} X_{(j, \bar{\mu})} \right) \right) \wedge \left(\bigwedge_{A \in \mathcal{A}^-} \neg \text{acy} \left(\bigcup_{j \in A} \bigcup_{\bar{\mu} \in [0, d]^{qt}} X_{(j, \bar{\mu})} \right) \right).$

Por definición, φ_K expresa la existencia de conjuntos $X_{T \in \mathcal{T}}$ tales que se satisfacen las subfórmulas $\text{part}(X_{T \in \mathcal{T}})$, $\text{proper}(X_{T \in \mathcal{T}})$, $\text{neigh}(X_{T \in \mathcal{T}})$, $\text{size}(X_{T \in \mathcal{T}})$, $\text{conn}(X_{T \in \mathcal{T}})$ y $\text{acyc}(X_{T \in \mathcal{T}})$. Como mencionamos arriba, $\text{part}(X_{T \in \mathcal{T}})$ garantiza que los conjuntos $X_{T \in \mathcal{T}}$ conforman una partición de $V(G)$. La subfórmula $\text{proper}(X_{T \in \mathcal{T}})$ expresa la condición 1 para todos los vértices. Por definición de $t_{r, \mu}(X)$, tenemos que la subfórmula $\text{neigh}(X_{T \in \mathcal{T}})$ asegura que para cada $(j, \bar{\mu}) \in \mathcal{T}$ tenemos $X_{(j, \bar{\mu})} \subseteq \bigcap_{i \in [1, t]} \bigcap_{h \in [1, q]} \{v \in V(G) : |N_G^{r_i}(v) \cap \left(\bigcup_{\bar{\mu}' \in [0, d]^{qt}} X_{(h, \bar{\mu}')} \right)| \in f(\bar{\mu}_h^i)\}$. En otras palabras, asegura que todo vértice $v \in X_{(j, \bar{\mu})}$ es tal que $|N_G^{r_i}(v) \cap \left(\bigcup_{\bar{\mu}' \in [0, d]^{qt}} X_{(h, \bar{\mu}')} \right)| \in f(\bar{\mu}_h^i)$ para todo $i \in [1, t]$ y todo $h \in [1, q]$, lo cual implica la condición 2. Finalmente, $\text{size}(X_{T \in \mathcal{T}})$, $\text{conn}(X_{T \in \mathcal{T}})$ y $\text{acyc}(X_{T \in \mathcal{T}})$ indican que las propiedades globales de tamaño, conectividad y aciclicidad son satisfechas (condiciones 3 a 5).

Notemos que $\mathcal{T}$ tiene $q(d+1)^{qt}$ elementos. Luego, la longitud de la fórmula $\text{part}(X_{T \in \mathcal{T}})$ es $O(q^2(d+1)^{2qt})$ y la longitud de $\text{proper}(X_{T \in \mathcal{T}})$ es $O(q(d+1)^{qt})$. Como la longitud de un término $t_{r_i, f(\bar{\mu}_h^i)}(X)$ es $O(d \cdot \text{length}(X))$, la longitud de $\text{neigh}(X_{T \in \mathcal{T}})$ es $O(tq^2d(d+1)^{2qt})$. Las longitudes de $\text{size}(X_{T \in \mathcal{T}})$, $\text{conn}(X_{T \in \mathcal{T}})$ y $\text{acyc}(X_{T \in \mathcal{T}})$ son, respectivamente, $O(q(d+1)^{qt})$, $O((|\mathcal{C}^+| + |\mathcal{C}^-|)q(d+1)^{qt})$ y $O((|\mathcal{A}^+| + |\mathcal{A}^-|)q(d+1)^{qt})$, ya que hay a lo sumo $q(d+1)^{qt}$ posibles tuplas $(j, \bar{\mu})$ y para las últimas dos fórmulas tenemos la conjunción de $|\mathcal{C}^+| + |\mathcal{C}^-|$ y $|\mathcal{A}^+| + |\mathcal{A}^-|$ términos de longitud $O(q(d+1)^{qt})$ cada uno. Por lo tanto, y como $|\mathcal{C}^+| + |\mathcal{C}^-| \leq 2^q$ y $|\mathcal{A}^+| + |\mathcal{A}^-| \leq 2^q$, la longitud de la fórmula completa es $O(tq^2d(d+1)^{2qt})$. $\square$

De acuerdo a [26, Teorema 1.2], cada uno de los problemas asociados a las fórmulas $\varphi_1, \dots, \varphi_{|\mathcal{K}|}$ de un problema Π como en el lema anterior, puede ser resuelto en tiempo $n^{O(wt^2q^4d^3(d+1)^{4qt})}$, donde w es la mim-width de un árbol binario de descomposición dado. Como $|\mathcal{K}| \leq n^{q-1}$, tenemos que resolver no más de n^{q-1} problemas. Luego buscamos la solución óptima entre todas las $|\mathcal{K}|$ soluciones obtenidas. Así, la complejidad total

de resolver el problema II de esta manera es $n^{O(wt^2q^4d^3(d+1)^{4qt})}$. Gracias a esta transformación podemos extender el resultado del Teorema 5.1.5 con propiedades globales de aciclicidad.

Teorema 8.3.3 ([103]). *Sean d, q y t enteros positivos. Para un problema localmente verificable con q colores, funciones de verificación y de peso que son d -estables a t distancias, y propiedades globales de tamaño, conectividad y aciclicidad, existe un algoritmo que resuelve el problema en grafos con un árbol binario de descomposición dado en tiempo $n^{O(wt^2q^4d^3(d+1)^{4qt})}$, donde n es la cantidad de vértices del grafo de entrada y w es la *mim-width* del árbol binario de descomposición asociado.*

Queda abierta la pregunta de si es posible expresar, en lógica A&C DN, propiedades globales de composición de condiciones locales donde el tamaño del monoide es polinomial y la función asociada es d -estable. De no ser posible, el resultado de la Sección 5.1.2 sería más general que los resultados de [26] salvo por las propiedades de aciclicidad.

Capítulo 9

Conclusiones y trabajo futuro

Presentamos un nuevo marco para formalizar lo que intuitivamente conocemos como *problemas localmente verificables*. Este marco nos permite estudiar bajo qué condiciones estos problemas son resolubles eficientemente en distintas clases de grafos. En particular, estudiamos las condiciones para obtener algoritmos FPT y XP parametrizados por treewidth (Capítulo 3), por treewidth más grado o, equivalentemente, por carvingwidth (Sección 3.3.1), clique-width (Capítulo 4) y mim-width (Capítulo 5).

La Tabla 9.1 resume los resultados obtenidos para problemas 1-localmente verificables sin propiedades globales ni etiquetas en las aristas. Los algoritmos principales presentados pueden extenderse con distintas *propiedades globales*, agrandando así la familia de problemas que podemos resolver. Estudiamos separadamente el caso en el que la distancia es mayor a 1 ($r > 1$) ya que el salto de complejidad puede ser grande (pudiendo llegar a ser para-NP-difícil si no agregamos restricciones adicionales). Curiosamente, uno de estos casos derivó en un nuevo resultado estructural relacionando treewidth, grado y potencia de un grafo (Lema 1.6.9).

En esta tesis se incluyeron, además, nuevos resultados sobre esquemas de aproximación polinomial para problemas localmente verificables en grafos planares (Capítulo 6). Ellos están basados en la técnica de Baker, utilizando los resultados obtenidos para treewidth del Capítulo 3. Existen otros esquemas de aproximación que se podrían intentar adaptar a nuestro marco de problemas localmente verificables (ver, por ejemplo, [89, 90]) y que ciertamente resultan atractivos para investigar en el futuro.

Como vimos en el Capítulo 8, todos estos resultados generalizan los obtenidos con los marcos de problemas LCC, ECC, LCVS y LCVP. En cuanto a la lógica A&C DN, pudimos probar que es equivalente bajo las condiciones impuestas para mim-width sin incluir propiedades globales de aciclicidad ni de composición de condiciones locales.

Nuestros resultados nos permitieron analizar la complejidad de problemas ya existentes en la literatura, para los cuales no se conocía su complejidad parametrizada por treewidth, clique-width y/o mim-width, y para algunos de los cuales tampoco se conocía un algoritmo aproximado para grafos planares. La Tabla 9.2 muestra un resumen de todo lo anteriormente mencionado.

Todos aquellos casos indicados con “?” en las Tablas 9.1 y 9.2 quedan abiertos para futuras investigaciones. Además, también quedan abiertas las siguientes preguntas:

- ¿Los problemas m -estables son XP parametrizados por mim-width? Esta pregunta se encuentra abierta incluso para el problema de dominación impar en grafos de intervalo propio. Conjeturamos que existe un problema en este grupo que es

para-NP-difícil parametrizado por *mim-width*.

- Los resultados XP (no FPT) obtenidos son ajustados en el sentido de que existen problemas $W[1]$ -difíciles en dichos grupos, pero ¿puede mejorarse significativamente la complejidad obtenida? Esto implica, por supuesto, un análisis de complejidad más fino.

Otra posible continuación natural de este trabajo es considerar otros parámetros de anchura en grafos, tales como los recientemente definidos *sim-width* [124], *one-sided-mim-width* [28], *neighbor-depth* [28], *twin-width* [38], *stretch-width* [37] o *tree-independence number* [72], analizando en cada caso qué restricciones son suficientes y/o necesarias para poder desarrollar algoritmos eficientes. Incluso resulta interesante analizar qué ocurre con el parámetro *carvingwidth*, ya que los resultados obtenidos fueron a partir del algoritmo de *treewidth*, y quizás un algoritmo que aproveche las propiedades de este parámetro nos permita obtener mejores resultados.

<i>check</i> , <i>w</i>	COLORS		
	Constante	Polinomial	No acotada
Computables en tiempo polinomial	FPT ⁽¹⁾ carw ? tw para-NP-h cw para-NP-d mimw	XP carw para-NP-d tw para-NP-d cw para-NP-d mimw	para-NP-d carw para-NP-d tw para-NP-d cw para-NP-d mimw
Sistema de vecindades parciales polinomial	FPT ⁽¹⁾ carw W[1]-h, XP tw para-NP-h cw para-NP-d mimw	XP carw W[1]-h, XP tw para-NP-d cw para-NP-d mimw	para-NP-d carw para-NP-d tw para-NP-d cw para-NP-d mimw
Sistema de vecindades parciales univ. polinomial	FPT ⁽¹⁾ carw W[1]-h, XP tw W[1]-h, XP cw para-NP-d mimw	XP carw W[1]-h, XP tw W[1]-h, XP cw para-NP-d mimw	? carw ? tw ? cw para-NP-d mimw
Sistema de vecindades parciales constante	FPT lineal carw FPT lineal tw para-NP-h cw para-NP-h mimw	XP carw W[1]-h, XP tw para-NP-d cw para-NP-d mimw	para-NP-d carw para-NP-d tw para-NP-d cw para-NP-d mimw
Sistema de vecindades parciales univ. constante	FPT lineal carw FPT lineal tw FPT ⁽²⁾ cw ? mimw	FPT carw FPT tw FPT cw ? mimw	? carw ? tw ? cw ? mimw
Contadora de colores	FPT ⁽¹⁾ carw W[1]-h, XP tw W[1]-h, XP cw para-NP-d mimw	XP carw para-NP-d tw para-NP-d cw para-NP-d mimw	? carw para-NP-d tw para-NP-d cw para-NP-d mimw
<i>m</i> -modular	FPT lineal carw FPT lineal tw FPT ⁽²⁾ cw ? mimw	XP carw ? tw para-NP-d cw para-NP-d mimw	? carw ? tw para-NP-d cw para-NP-d mimw
<i>d</i> -estable	FPT lineal carw FPT lineal tw FPT ⁽²⁾ cw W[1]-h, XP ⁽³⁾ mimw	XP carw para-NP-d tw para-NP-d cw para-NP-d mimw	? carw para-NP-d tw para-NP-d cw para-NP-d mimw

Tabla 9.1: Resumen de algunos de los resultados principales sobre la complejidad de problemas 1-localmente verificables bajo distintas restricciones sobre la cantidad de colores y las funciones de verificación y peso. Aquí, cuando indicamos para-NP-h (o W[1]-h), estamos diciendo que existe un problema con esas condiciones que es para-NP-h (o W[1]-h) parametrizado por el parámetro correspondiente. (1) FPT lineal si *check* y *w* son computables en tiempo constante. (2) FPT lineal si una expresión clique-width adecuada viene dada. (3) Dado un árbol binario de descomposición adecuada.

Problema	tw	cw	mimw	PTAS
$[k]$ -dominación romana	FPT-L	FPT ^(*)	XP	sí
$[k]$ -dominación romana total	FPT-L	FPT ^(*)	XP	sí
Dominación romana perfecta (doble)	FPT-L	FPT ^(*)	XP	?
$[k]$ -dominación romana independiente	FPT-L	FPT ^(*)	XP	?
$[k]$ -dominación romana indep. exterior	FPT-L	FPT ^(*)	XP	sí
$[k]$ -dominación romana maximal	FPT-L	FPT ^(*)	XP	?
Dominación romana con signo	XP	XP	?	?
Dominación romana total con signo	XP	XP	?	?
$[k]$ -dominación romana global	XP	XP	?	?
$\{k\}$ -dominación romana	FPT-L	FPT ^(*)	XP	sí
Máxima dominación dual acotada	FPT-L	FPT ^(*)	XP	?
Máxima dominación dual	FPT-L	FPT ^(*)	XP	?
Mínima dominación dual negativa	FPT-L	FPT ^(*)	XP	?
Máximo conjunto de vértices k -felices	FPT-L	FPT ^(*) [31]	XP	sí
Máximo conjunto feliz	FPT-L	FPT ^(*)	XP	?
Coloreo localmente acotado	FPT-L[24]	FPT ^(*)	XP	?
k -coloreo libre de conflictos	FPT-L	FPT ^(*) (1)	XP ⁽²⁾	?
M -GPF	FPT-L	FPT ^(*) [113]	XP	?
GPF con k y u polinomiales	XP	XP	?	?
k -comunidad	XP	XP	?	?
k -comunidad balanceada	XP	XP	?	?
k -comunidad débil	XP	XP	?	?
Max-PDS	XP	XP	?	?
Extensión PDS	XP	XP	?	?
Máxima γ -quasi-clique basada en grados	XP	XP	?	?
Máxima γ -quasi-clique	XP	XP	?	?
b-coloreo, con número fijo de colores	FPT-L ⁽³⁾	FPT ^(*) (4)	XP ⁽²⁾	?
Coloreo aditivo	XP	?	?	?
Mínima violación cromática	XP	?	?	?
Dominación Grundy	XP	?	?	sí
Dominación Grundy total	XP	?	?	sí

Tabla 9.2: Listado de nuevos resultados respecto a problemas ya existentes y su complejidad parametrizada según la treewidth (tw) del grafo, la clique-width (cw) del grafo y la mim-width (mimw) de un árbol binario de descomposición dado. También se indica si admiten un PTAS en grafos planares. FPT-L significa “FPT lineal”. (*) FPT lineal si ya contamos con una expresión clique-width adecuada. (1) Este resultado también fue obtenido independientemente en [30]. (2) Estos resultados también fueron obtenidos independientemente en [26]. (3) Este resultado también fue obtenido independientemente en [12]. (4) En [122] presentan un algoritmo XP parameterizado por clique-width para b-coloreo sin restricciones en la cantidad de colores, y en [123] prueban que este problema es W[1]-h parameterizado por pathwidth.

Apéndice A

Definiciones de problemas

CONJUNTO DOMINANTE [147]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que $|N[v] \cap S| \geq 1$ para cada $v \in V(G)$?

DOMINACIÓN TOTAL [61]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que $|N(v) \cap S| \geq 1$ para cada $v \in V(G)$?

DOMINACIÓN k -UPLA [107]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que $|N[v] \cap S| \geq k$ para cada $v \in V(G)$?

DOMINACIÓN k -UPLA TOTAL [111]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que $|N(v) \cap S| \geq k$ para cada $v \in V(G)$?

k -DOMINACIÓN [84]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que $|N(v) \cap S| \geq k$ para cada $v \in V(G) - S$?

$\{k\}$ -DOMINACIÓN [109]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{0, 1, \dots, k\}$ de peso a lo sumo ℓ y tal que $f(N[v]) \geq k$ para cada $v \in V(G)$?

DOMINACIÓN k -ARCOÍRIS [56]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow 2^{\{1, \dots, k\}}$ de peso $(\sum_{v \in V(G)} |f(v)|)$ a lo sumo ℓ y tal que para cada vértice $v \in V(G)$ para el cual $f(v) = \emptyset$ tenemos que $\bigcup_{u \in N_G[v]} f(u) = \{1, \dots, k\}$?

DOMINACIÓN SEMITOTAL [98]

Instancia: Un grafo G (con pesos) sin vértices aislados y un entero positivo ℓ .

Pregunta: ¿Existe un conjunto dominante $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que cada vértice en S está a distancia a lo sumo 2 de otro vértice en S ?

DOMINACIÓN A DISTANCIA k [110, 141]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que cada vértice en G está a distancia a lo sumo k de un vértice en S ?

CONJUNTO DOMINANTE CONEXO [154]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe un conjunto dominante de G de tamaño (peso) a lo sumo ℓ tal que induce un subgrafo conexo en G ?

DOMINACIÓN ROMANA [62]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{0, 1, 2\}$ de peso a lo sumo ℓ y tal que cada vértice $u \in V(G)$ para el cual $f(u) = 0$ es adyacente a al menos un vértice $v \in V(G)$ para el cual $f(v) = 2$?

 $[k]$ -DOMINACIÓN ROMANA TOTAL [6, 157]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $[k]$ -dominante romana $f: V(G) \rightarrow \llbracket 0, k+1 \rrbracket$ de peso a lo sumo ℓ y tal que el conjunto de vértices con valores positivos no tenga vértices aislados?

 $[k]$ -DOMINACIÓN ROMANA INDEPENDIENTE [6, 137]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $[k]$ -dominante romana de peso a lo sumo ℓ y tal que el conjunto de vértices con valores positivos sea independiente?

 $[k]$ -DOMINACIÓN ROMANA INDEPENDIENTE EXTERIOR [3, 6]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $[k]$ -dominante romana de peso a lo sumo ℓ y tal que el conjunto de vértices con valor 0 sea independiente?

DOMINACIÓN ROMANA PERFECTA [18]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{0, 1, 2\}$ de peso a lo sumo ℓ y tal que cada vértice $u \in V(G)$ para el cual $f(u) = 0$ es adyacente a exactamente un vértice $v \in V(G)$ para el cual $f(v) = 2$?

DOMINACIÓN ROMANA PERFECTA DOBLE [75]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{0, 1, 2, 3\}$ de peso a lo sumo ℓ y tal que para cada vértice $u \in V(G)$ se cumple lo siguiente: (i) si $f(u) = 0$ entonces o bien u es adyacente a exactamente dos vértices de valor 2 y ninguno de valor 3, o bien es adyacente a exactamente un vértice de valor 3 y ninguno de valor 2; (ii) si $f(u) = 2$ entonces u es adyacente a exactamente un vértice de valor 2 y ninguno de valor 3?

k-DOMINACIÓN ROMANA CON SIGNO [5]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{-1, 1, 2\}$ de peso a lo sumo ℓ y tal que, para cada vértice $v \in V(G)$, $f(N[v]) \geq 1$ y si $f(v) = -1$ entonces v es adyacente a al menos un vértice $u \in V(G)$ con $f(u) = 2$?

k-DOMINACIÓN ROMANA TOTAL CON SIGNO [172]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{-1, 1, 2\}$ de peso a lo sumo ℓ y tal que, para cada vértice $v \in V(G)$, $f(N(v)) \geq 1$ y si $f(v) = -1$ entonces v es adyacente a al menos un vértice $u \in V(G)$ con $f(u) = 2$?

DOMINACIÓN GRUNDY TOTAL [47]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una sucesión $(v_1, \dots, v_k)$ de vértices distintos de G tal que $k \geq \ell$, $\{v_1, \dots, v_k\}$ es un conjunto dominante de G y $N(v_i) - \bigcup_{j=1}^{i-1} N(v_j) \neq \emptyset$ para cada $i \in [1, k]$?

k-COLOREO [92]

Instancia: Un grafo G .

Pregunta: ¿Existe una función $c: V(G) \rightarrow \{1, \dots, k\}$ tal que $c(u) \neq c(v)$ para toda arista $uv \in E(G)$?

COLOREO POR LISTAS [76, 171]

Instancia: Un grafo G y un conjunto $L(v)$ de colores para cada vértice $v \in V(G)$.

Pregunta: ¿Existe una función $c: V(G) \rightarrow \bigcup_{v \in V(G)} L_v$ tal que $c(u) \neq c(v)$ para toda arista $uv \in E(G)$ y $c(v) \in L_v$ para todo vértice $v \in V(G)$?

COLOREO ADITIVO [70]

Instancia: Un grafo G y un entero positivo k .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{1, \dots, k\}$ tal que para cada par de vértices adyacentes u, v las sumas de los números asignados a sus vecinos son diferentes (es decir, $\sum_{w \in N(u)} f(w) \neq \sum_{z \in N(v)} f(z)$)?

COLOREO ACÍCLICO [105]

Instancia: Un grafo G y un entero positivo k .

Pregunta: ¿Existe un k -coloreo de G tal que cada subgrafo de G inducido por los vértices de dos de los colores es acíclico (en otras palabras, un bosque)?

ETIQUETADO $L(h, k)$ [51]

Instancia: Un grafo G y un entero positivo s .

Pregunta: ¿Existe un etiquetado de los vértices de G con enteros entre 0 y s tal que vértices adyacentes de G son etiquetados usando enteros con diferencia al menos h , y vértices con un vecino en común son etiquetados usando enteros con diferencia al menos k ?

CONJUNTO INDEPENDIENTE [92]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe un conjunto independiente en G de tamaño (peso) al menos ℓ ?

CONJUNTO INDEPENDIENTE A DISTANCIA k [78, 85]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe $X \subseteq V(G)$ de tamaño al menos ℓ tal que la distancia entre cada par de vértices de X es al menos $k + 1$?

FUNCIÓN DE $\{k\}$ -EMPAQUETAMIENTO [133]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe una función $f: V(G) \rightarrow \{0, 1, \dots, k\}$ de peso al menos ℓ y tal que $f(N[v]) \leq k$ para cada $v \in V(G)$?

 k -EMPAQUETAMIENTO LIMITADO [91]

Instancia: Un grafo G y un entero positivo ℓ .

Pregunta: ¿Existe $B \subseteq V(G)$ de tamaño al menos ℓ y tal que $|N[v] \cap B| \leq k$ para cada $v \in V(G)$?

NÚMERO CROMÁTICO DE EMPAQUETAMIENTO [48]

Instancia: Un grafo G y un entero positivo k .

Pregunta: ¿Puede G ser partido en clases disjuntas $X_1, \dots, X_k$ donde, para todo $i \in \llbracket 1, k \rrbracket$, cada par de vértices en X_i tiene distancia mayor a i ?

EMPAREJAMIENTO [74]

Instancia: Un grafo G (con pesos en las aristas) y un entero positivo ℓ .

Pregunta: ¿Existe un conjunto $M \subseteq E(G)$ de tamaño (peso) al menos ℓ y tal que cada par de aristas en M no comparte ningún extremo?

DOMINACIÓN POR ARISTAS [174]

Instancia: Un grafo G (con pesos en las aristas) y un entero positivo ℓ .

Pregunta: ¿Existe $F \subseteq E(G)$ de tamaño (peso) a lo sumo ℓ y tal que cada arista en $E(G)$ comparte un extremo con al menos una arista en F ?

CUBRIMIENTO POR VÉRTICES [125]

Instancia: Un grafo G (con pesos) y un entero positivo ℓ .

Pregunta: ¿Existe $S \subseteq V(G)$ de tamaño (peso) a lo sumo ℓ y tal que cada arista en $E(G)$ tiene al menos un extremo en S ?

CUBRIMIENTO POR ARISTAS [92]

Instancia: Un grafo G (con pesos en las aristas) y un entero positivo ℓ .

Pregunta: ¿Existe $F \subseteq E(G)$ de tamaño (peso) a lo sumo ℓ y tal que cada vértice en $V(G)$ pertenece a al menos una arista en F ?

Apéndice B

Abreviaturas

B.1. Complejidad

FPT	Tratable de parámetro fijo (fixed-parameter tractable). Sección 1.1.1.
NP-c	NP-completo. Sección 1.1.
NP-h	NP-difícil (NP-hard). Sección 1.1.
para-NP-h	Para-NP-difícil (para-NP-hard). Sección 1.1.1.
W[1]-h	W[1]-difícil (W[1]-hard). Sección 1.1.1.
XP	Polinomial en rebanadas (slice-wise polynomial). Sección 1.1.1.

B.2. Parámetros de anchura

carw	Carvingwidth. Sección 1.6.3.
cutw	Cutwidth. Sección 1.6.3.
cw	Clique-width. Sección 1.6.2.
lmimw	Mim-width lineal (linear mim-width). Sección 1.6.3.
lsimw	Sim-width lineal (linear sim-width). Sección 1.6.3.
mimw	Mim-width. Sección 1.6.3.
mmw	MM-width. Sección 1.6.3.
pw	Pathwidth. Sección 1.6.1.
rw	Rankwidth. Sección 1.6.3.
simw	Sim-width. Sección 1.6.3.
tw	Treewidth. Sección 1.6.1.

B.3. Marcos de problemas

A&C DN	Lógica de vecindad a distancia con restricciones de aciclicidad y conectividad (distance neighborhood logic with acyclicity and connectivity constraints). Sección 8.3.
ECC	Problemas de composición de condiciones locales (edge condition composition). Sección 8.2.
LCC	Problemas de composición de condiciones en aristas (local condition composition). Sección 8.2.
LCVP	Problemas localmente verificables de partición de vértices (locally checkable vertex partitioning). Sección 8.1.
LCVS	Problemas localmente verificables de subconjuntos de vértices (locally checkable vertex subset). Sección 8.1.

B.4. Otros

PTAS	Esquema de aproximación de tiempo polinomial (polynomial-time approximation scheme). Capítulo 6.
-------------	--

Apéndice C

Resumen de nuestro marco

C.1. Elementos

LABELS	Conjunto de etiquetas para aristas.
$\ell_e \in \text{LABELS}$	Para cada arista e , una etiqueta.
COLORS	Conjunto de colores.
$L_v \subseteq \text{COLORS}$	Para cada vértice v , un conjunto no vacío de posibles colores.
$(\text{WEIGHTS}, \preceq, \oplus)$	Conjunto de pesos. Ver Sección 1.4.
$w(v, c)$	Función de peso. Entrada: vértice v y coloreo c de la vecindad cerrada de v . Salida: peso en WEIGHTS .
$check(v, c)$	Función de verificación, propiedad local. Entrada: vértice v y coloreo c de la vecindad cerrada de v . Salida: TRUE o FALSE .

C.1.1. Propiedades globales

Tamaño	Sección 2.3.1.
Conectividad (positiva, negativa)	Sección 2.3.2.
Aciclicidad (positiva, negativa)	Sección 2.3.3.
Composición de condiciones locales	Sección 2.3.4.

C.2. Restricciones

Sistema de vecindades parciales (polinomial, constante, universal)	Sección 2.4.1.
Funciones contadoras de colores	Sección 2.4.2.
Funciones d -estables	Sección 2.4.3.
Funciones m -modulares	Sección 2.4.4.
Jerarquía de colores	Sección 6.2.
Colores neutrales	Sección 6.2.

Bibliografía

- [1] H. Abdollahzadeh Ahangar, A. Bahremandpour, S. M. Sheikholeslami, N. D. Soner, Z. Tahmasbzadehbaee, and L. Volkmann. Maximal Roman domination numbers in graphs. *Utilitas Mathematica*, 103:245–258, 02 2017.
- [2] H. Abdollahzadeh Ahangar, M. Chellali, and S. M. Sheikholeslami. On the double Roman domination in graphs. *Discrete Applied Mathematics*, 232:1–7, 2017.
- [3] H. Abdollahzadeh Ahangar, M. Chellali, and S. M. Sheikholeslami. Outer independent double Roman domination. *Applied Mathematics and Computation*, 364:124617, 2020.
- [4] H. Abdollahzadeh Ahangar, M. Chellali, S. M. Sheikholeslami, and J. C. Valenzuela-Tripodoro. Maximal double Roman domination in graphs. *Applied Mathematics and Computation*, 414:126662, 2022.
- [5] H. Abdollahzadeh Ahangar, M. A. Henning, C. Löwenstein, Y. Zhao, and V. Samodivkin. Signed Roman domination in graphs. *J. Comb. Optim.*, 27(2):241–255, feb 2014.
- [6] H. Abdollahzadeh Ahangar, M. P. Álvarez, M. Chellali, S. M. Sheikholeslami, and J. C. Valenzuela-Tripodoro. Triple Roman domination in graphs. *Applied Mathematics and Computation*, 391:125444, 2021.
- [7] Z. Abel, V. Alvarez, E. D. Demaine, S. P. Fekete, A. Gour, A. Hesterberg, P. Keldenich, and C. Scheffer. Conflict-free coloring of graphs. *SIAM Journal on Discrete Mathematics*, 32(4):2675–2702, 2018.
- [8] R. Adhikary, K. Bose, S. Mukherjee, and B. Roy. Complexity of Maximum Cut on Interval Graphs. In K. Buchin and E. Colin de Verdière, editors, *37th International Symposium on Computational Geometry (SoCG 2021)*, volume 189 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:11, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [9] A. Agrawal, N. R. Aravind, S. Kalyanasundaram, A. S. Kare, J. Lauri, N. Misra, and I. V. Reddy. Parameterized complexity of happy coloring problems. *Theoretical Computer Science*, 835:58–81, 2020.
- [10] A. Agrawal, P. Choudhary, N. S. Narayanaswamy, K. K. Nisha, and V. Ramamoorthi. Parameterized complexity of minimum membership dominating set. *Algorithmica*, 85:3430–3452, 2023.

- [11] S. Akbari, M. Ghanbari, R. Manaviyat, and S. Zare. On the lucky choice number of graphs. *Graphs and Combinatorics*, 29(2):157–163, Mar. 2013.
- [12] D. Andrade and A. Silva. (Sub)fall coloring and b-coloring parameterized by treewidth. In *Anais do VII Encontro de Teoria da Computação*, pages 69–72, Porto Alegre, RS, Brasil, 2022. SBC.
- [13] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k -trees. *Discrete Applied Mathematics*, 23(1):11–24, 1989.
- [14] Y. Asahiro, H. Eto, T. Hanaka, G. Lin, E. Miyano, and I. Terabaru. Complexity and approximability of the happy set problem. *Theoretical Computer Science*, 866:123–144, 2021.
- [15] Y. Asahiro, H. Eto, T. Hanaka, G. Lin, E. Miyano, and I. Terabaru. Parameterized algorithms for the happy set problem. *Discrete Applied Mathematics*, 304:32–44, 2021.
- [16] N. Baghirova, C. L. Gonzalez, B. Ries, and D. Schindl. Locally Checkable Problems Parameterized by Clique-Width. In S. W. Bae and H. Park, editors, *33rd International Symposium on Algorithms and Computation (ISAAC 2022)*, volume 248 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 31:1–31:20, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [17] B. S. Baker. Approximation Algorithms for NP-Complete Problems on Planar Graphs. *J. ACM*, 41(1):153–180, jan 1994.
- [18] S. Banerjee, J. Mark Keil, and D. Pradhan. Perfect Roman domination in graphs. *Theoretical Computer Science*, 796:1–21, 2019.
- [19] C. Bazgan, J. Chlebikova, and T. Pontoizeau. Structural and algorithmic properties of 2-community structures. *Algorithmica*, 80:1890–1908, 2018.
- [20] C. Bazgan, J. Chlebíková, C. Dallard, and T. Pontoizeau. Proportionally dense subgraph of maximum size: Complexity and approximation. *Discrete Applied Mathematics*, 270:25–36, 2019.
- [21] R. A. Beeler, T. W. Haynes, and S. T. Hedetniemi. Double Roman domination. *Discrete Applied Mathematics*, 211:23–29, 2016.
- [22] R. Belmonte, P. van ’t Hof, M. Kamiński, D. Paulusma, and D. M. Thilikos. Characterizing graphs of small carving-width. *Discrete Applied Mathematics*, 161(13):1888–1893, 2013.
- [23] R. Belmonte and M. Vatshelle. Graph classes with structured neighborhoods and algorithmic applications. *Theoretical Computer Science*, 511:54–65, 2013. Exact and Parameterized Computation.
- [24] C. Bentz. Weighted and locally bounded list-colorings in split graphs, cographs, and partial k -trees. *Theoretical Computer Science*, 782:11–29, 2019.

- [25] C. Bentz and C. Picouleau. Locally bounded k -colorings of trees. *RAIRO-Operations Research*, 43(1):27–33, 2009.
- [26] B. Bergougnoux, J. Dreier, and L. Jaffke. A logic-based algorithmic meta-theorem for mim-width. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3282–3304.
- [27] B. Bergougnoux and M. M. Kanté. More applications of the d -neighbor equivalence: Acyclicity and connectivity constraints. *SIAM Journal on Discrete Mathematics*, 35(3):1881–1926, 2021.
- [28] B. Bergougnoux, T. Korhonen, and I. Razgon. New Width Parameters for Independent Set: One-Sided-Mim-Width and Neighbor-Depth. In D. Paulusma and B. Ries, editors, *Graph-Theoretic Concepts in Computer Science*, pages 72–85, Cham, 2023. Springer Nature Switzerland.
- [29] U. Bertelé and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, Inc., USA, 1972.
- [30] S. Bhyravarapu, T. A. Hartmann, S. Kalyanasundaram, and I. Vinod Reddy. Conflict-free coloring: Graphs of bounded clique width and intersection graphs. In P. Flocchini and L. Moura, editors, *Combinatorial Algorithms*, pages 92–106, Cham, 2021. Springer International Publishing.
- [31] I. Bliznets and D. Sagunov. Maximizing happiness in graphs of bounded clique-width. In Y. Kohayakawa and F. K. Miyazawa, editors, *LATIN 2020: Theoretical Informatics*, pages 91–103, Cham, 2020. Springer International Publishing.
- [32] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. Technical report, USA, 1987.
- [33] H. L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In T. Lepistö and A. Salomaa, editors, *Automata, Languages and Programming*, pages 105–118, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
- [34] H. L. Bodlaender. A partial k -arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1–45, 1998.
- [35] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015. 40th International Colloquium on Automata, Languages and Programming (ICALP 2013).
- [36] H. L. Bodlaender, P. G. Drange, M. S. Dregi, F. V. Fomin, D. Lokshtanov, and M. Pilipczuk. An $O(c^k n)$ 5-approximation algorithm for treewidth. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 499–508, 2013.
- [37] E. Bonnet and J. Duron. Stretch-width. 2023. arXiv e-prints arXiv:2305.12023 [cs.DM].

- [38] E. Bonnet, E. J. Kim, S. Thomassé, and R. Watrigant. Twin-width I: Tractable FO model checking. *J. ACM*, 69(1), nov 2021.
- [39] F. Bonomo and D. de Estrada. On the thinness and proper thinness of a graph. *Discrete Applied Mathematics*, 261:78–92, 2019. GO X Meeting, Rigi Kaltbad (CH), July 10–14, 2016.
- [40] F. Bonomo, L. N. Grippio, M. Milanič, and M. D. Safe. Graph classes with and without powers of bounded clique-width. *Discrete Applied Mathematics*, 199:3–15, 2016. Sixth Workshop on Graph Classes, Optimization, and Width Parameters, Santorini, Greece, October 2013.
- [41] F. Bonomo-Braberman and C. L. Gonzalez. A new approach on locally checkable problems. *Discrete Applied Mathematics*, 314:53–80, 2022.
- [42] K. S. Booth and J. H. Johnson. Dominating sets in chordal graphs. *SIAM Journal on Computing*, 11(1):191–199, 1982.
- [43] R. C. Bottesch. On W[1]-Hardness as Evidence for Intractability. In I. Potapov, P. Spirakis, and J. Worrell, editors, *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [44] M. Braga, D. Delle Donne, M. Escalante, J. Marenco, M. E. Ugarte, and M. C. Varaldo. The minimum chromatic violation problem: a polyhedral study. *Electronic Notes in Discrete Mathematics*, 62:309–314, 2017. LAGOS’17 – IX Latin and American Algorithms, Graphs and Optimization.
- [45] A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*, volume 3 of *SIAM Monographs on Discrete Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 1999.
- [46] B. Brešar, T. Gologranc, M. Milanič, D. F. Rall, and R. Rizzi. Dominating sequences in graphs. *Discrete Mathematics*, 336:22–36, 2014.
- [47] B. Brešar, M. A. Henning, and D. F. Rall. Total dominating sequences in graphs. *Discrete Mathematics*, 339(6):1665–1676, 2016.
- [48] B. Brešar, S. Klavžar, and D. F. Rall. On the packing chromatic number of Cartesian products, hexagonal lattice, and trees. *Discrete Applied Mathematics*, 155(17):2303–2311, 2007.
- [49] N. Brettell, J. Horsfield, A. Munaro, G. Paesani, and D. Paulusma. Bounding the mim-width of hereditary graph classes. *Journal of Graph Theory*, 99(1):117–151, 2022.
- [50] B.-M. Bui-Xuan, J. A. Telle, and M. Vatshelle. Fast dynamic programming for locally checkable vertex subset and vertex partitioning problems. *Theoretical Computer Science*, 511:66–76, 2013.

- [51] T. Calamoneri. The $L(h, k)$ -Labelling Problem: A Survey and Annotated Bibliography. *The Computer Journal*, 49(5):585–608, 05 2006.
- [52] Y. Caro and M. S. Jacobson. On non- $z \pmod k$ dominating sets. *Discussiones Mathematicae Graph Theory*, 23(1):189–199, 2003.
- [53] D. Cattanéo and S. Perdrix. The parameterized complexity of domination-type problems and application to linear codes. In T. V. Gopal, M. Agrawal, A. Li, and S. B. Cooper, editors, *Theory and Applications of Models of Computation - 11th Annual Conference, TAMC 2014, Chennai, India, April 11-13, 2014. Proceedings*, volume 8402 of *Lecture Notes in Computer Science*, pages 86–103. Springer, 2014.
- [54] G. J. Chang and D. Kuo. The $L(2, 1)$ -labeling problem on graphs. *SIAM Journal on Discrete Mathematics*, 9(2):309–316, 1996.
- [55] G. J. Chang and C. Lu. Distance-two labelings of graphs. *European Journal of Combinatorics*, 24(1):53–58, 2003.
- [56] G. J. Chang, J. Wu, and X. Zhu. Rainbow domination on trees. *Discrete Applied Mathematics*, 158(1):8–12, 2010.
- [57] M. Chellali, T. W. Haynes, S. T. Hedetniemi, and A. A. McRae. Roman $\{2\}$ -domination. *Discrete Applied Mathematics*, 204:22–28, 2016.
- [58] N. Chiba, T. Nishizeki, S. Abe, and T. Ozawa. A linear algorithm for embedding planar graphs using pq-trees. *Journal of Computer and System Sciences*, 30(1):54–76, 1985.
- [59] F. R. K. Chung. On the cutwidth and the topological bandwidth of a tree. *SIAM Journal on Algebraic Discrete Methods*, 6(2):268–277, 1985.
- [60] E. Cockayne, S. Goodman, and S. Hedetniemi. A linear algorithm for the domination number of a tree. *Information Processing Letters*, 4(2):41–44, 1975.
- [61] E. J. Cockayne, R. M. Dawes, and S. T. Hedetniemi. Total domination in graphs. *Networks*, 10(3):211–219, 1980.
- [62] E. J. Cockayne, P. A. Dreyer, S. M. Hedetniemi, and S. T. Hedetniemi. Roman domination in graphs. *Discrete Mathematics*, 278(1):11–22, 2004.
- [63] S. A. Cook. 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, 1971. Association for Computing Machinery.
- [64] G. Cordasco, L. Gargano, and A. A. Rescigno. Dual domination problems in graphs. *Journal of Computer and System Sciences*, 128:18–34, 2022.
- [65] D. G. Corneil and U. Rotics. On the relationship between clique-width and treewidth. *SIAM Journal on Computing*, 34(4):825–847, 2005.
- [66] B. Courcelle, J. Engelfriet, and G. Rozenberg. Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences*, 46(2):218–270, 1993.

- [67] B. Courcelle, J. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory Comput. Systems*, 33:125–150, 2000.
- [68] B. Courcelle and M. Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1):49–82, 1993.
- [69] B. Courcelle and S. Olariu. Upper bounds to the clique width of graphs. *Discrete Applied Mathematics*, 101(1):77–114, 2000.
- [70] S. Czerwiński, J. Grytczuk, and W. Żelazny. Lucky labelings of graphs. *Information Processing Letters*, 109(18):1078–1081, 2009.
- [71] D. P. Dailey. Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete. *Discrete Mathematics*, 30(3):289–293, 1980.
- [72] C. Dallard, M. Milanič, and K. Štorgel. Treewidth versus clique number. II. Tree-independence number. *Journal of Combinatorial Theory, Series B*, 164:404–442, 2024.
- [73] R. G. Downey and M. R. Fellows. *Parameterized Complexity*, volume XV of *Monographs in Computer Science*. Springer, New York, NY, 1999.
- [74] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
- [75] A. T. Egunjobi and T. W. Haynes. Perfect double Roman domination of trees. *Discrete Applied Mathematics*, 284:71–85, 2020.
- [76] P. Erdos, A. L. Rubin, and H. Taylor. Choosability in graphs. In *Proc. West Coast Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, volume 26, pages 125–157, 1979.
- [77] V. Estivill-Castro and M. Parsa. On connected two communities. In *Proceedings of the Thirty-Sixth Australasian Computer Science Conference - Volume 135*, ACSC '13, pages 23–30, AUS, 2013. Australian Computer Society, Inc.
- [78] H. Eto, F. Guo, and E. Miyano. Distance- d independent set problems for bipartite and chordal graphs. *J. Comb. Optim.*, 27(1):88–99, Jan. 2014.
- [79] G. Even, Z. Lotker, D. Ron, and S. Smorodinsky. Conflict-free colorings of simple geometric regions with applications to frequency assignment in cellular networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 691–700, 2002.
- [80] M. R. Fellows, F. V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, and C. Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.
- [81] L. Fernández and V. Leoni. New complexity results on Roman $\{2\}$ -domination. *RAIRO - Operations Research*, 57(4):1905–1912, 2023.

- [82] J. Fiala and P. A. Golovach. Complexity of the packing coloring problem for trees. *Discrete Applied Mathematics*, 158(7):771–778, 2010. Third Workshop on Graph Classes, Optimization, and Width Parameters Eugene, Oregon, USA, October 2007.
- [83] J. Fiala, P. A. Golovach, and J. Kratochvíl. Distance constrained labelings of graphs of bounded treewidth. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming*, pages 360–372, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [84] J. F. Fink and M. S. Jacobson. *n-Domination in Graphs*, pages 283–300. John Wiley & Sons, Inc., USA, 1985.
- [85] P. Firby and J. Haviland. Independence and average distance in graphs. *Discrete Applied Mathematics*, 75(1):27–37, 1997.
- [86] F. V. Fomin, P. A. Golovach, D. Lokshantov, and S. Saurabh. Almost optimal lower bounds for problems parameterized by clique-width. *SIAM Journal on Computing*, 43(5):1541–1563, 2014.
- [87] F. V. Fomin, P. A. Golovach, and J.-F. Raymond. On the tractability of optimization problems on h -graphs. *Algorithmica*, 82:2432–2473, 2020.
- [88] F. V. Fomin and T. Korhonen. Fast FPT-approximation of branchwidth. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, pages 886–899, New York, NY, USA, 2022. Association for Computing Machinery.
- [89] E. Galby and A. Munaro. Approximating independent set and dominating set on VPG graphs, 2020. arXiv e-prints arXiv:2004.07566 [cs.DS].
- [90] E. Galby, A. Munaro, and S. Yang. Polynomial-Time Approximation Schemes for Independent Packing Problems on Fractionally Tree-Independence-Number-Fragile Graphs. In E. W. Chambers and J. Gudmundsson, editors, *39th International Symposium on Computational Geometry (SoCG 2023)*, volume 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [91] R. Gallant, G. Gunther, B. Hartnell, and D. Rall. Limited packings in graphs. *Discrete Applied Mathematics*, 158(12):1357–1364, 2010.
- [92] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
- [93] E. Gassner and J. Hatzl. A parity domination problem in graphs with bounded treewidth and distance-hereditary graphs. *Computing*, 82:171–187, 2008.
- [94] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

- [95] M. U. Gerber and D. Kobler. Algorithms for vertex-partitioning problems on graphs with fixed clique-width. *Theoretical Computer Science*, 299:719–734, 2003.
- [96] V. Giakoumakis, F. Roussel, and H. Thuillier. On P_4 -tidy graphs. *Discrete Mathematics & Theoretical Computer Science*, 1, 1 1997.
- [97] W. Goddard, S. M. Hedetniemi, S. T. Hedetniemi, J. M. Harris, and D. F. Rall. Broadcast chromatic numbers of graphs. *Ars Combinatoria*, 86:33–49, 2008.
- [98] W. Goddard, M. A. Henning, and C. A. McPillan. Semitotal domination in graphs. *Utilitas Mathematica*, 94:67–81, 2014.
- [99] M. C. Golumbic and U. Rotics. On the clique-width of perfect graph classes. In P. Widmayer, G. Neyer, and S. Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science*, pages 135–147, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
- [100] M. C. Golumbic and A. Sainte-Laguë. *The Zeroth Book of Graph Theory: An Annotated Translation of Les Réseaux (ou Graphes)—André Sainte-Laguë (1926)*, volume 2261 of *Lecture Notes in Mathematics*. Springer Cham, 1st edition, 2021.
- [101] D. Gonçalves. On the $L(p, 1)$ -labelling of graphs. In S. Felsner, editor, *2005 European Conference on Combinatorics, Graph Theory and Applications (EuroComb '05)*, volume DMTCS Proceedings vol. AE, European Conference on Combinatorics, Graph Theory and Applications (EuroComb '05) of *DMTCS Proceedings*, pages 81–86, Berlin, Germany, 2005. Discrete Mathematics and Theoretical Computer Science.
- [102] C. L. Gonzalez. El problema de dominación Grundy en grafos block y grafos araña bien etiquetada. Tesis de Licenciatura, Universidad Nacional de Rosario, 2017.
- [103] C. L. Gonzalez and F. Mann. On d -stable locally checkable problems parameterized by mim-width. *Discrete Applied Mathematics*, 347:1–22, 2024.
- [104] J. R. Griggs and R. K. Yeh. Labelling graphs with a condition at distance 2. *SIAM Journal on Discrete Mathematics*, 5(4):586–595, 1992.
- [105] B. Grünbaum. Acyclic colorings of planar graphs. *Israel Journal of Mathematics*, 14(4):390–408, 1973.
- [106] R. Halin. S -functions for graphs. *Journal of Geometry*, 8(1–2):171–186, 1976.
- [107] F. Harary and T. W. Haynes. Double domination in graphs. *Ars Combinatoria*, 55:201–213, 04 2000.
- [108] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Boca Raton: CRC Press, 1st edition, 1998.
- [109] J. He and H. Liang. Complexity of total $\{k\}$ -domination and related problems. In M. Atallah, X.-Y. Li, and B. Zhu, editors, *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, pages 147–155, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

- [110] M. A. Henning. Distance domination in graphs. In T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, editors, *Domination in Graphs: Advanced Topics*, chapter 12. Marcel Dekker, Inc., 1997.
- [111] M. A. Henning and A. P. Kazemi. k -tuple total domination in graphs. *Discrete Applied Mathematics*, 158(9):1006–1011, 2010.
- [112] M. A. Henning and W. F. Klostermeyer. Italian domination in trees. *Discrete Applied Mathematics*, 217:557–564, 2017.
- [113] E. Hinrichsen, G. Nasini, and N. Vansteenkiste. On general packing functions in graphs (brief announcement). *Procedia Computer Science*, 223:367–369, 2023. XII Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 2023).
- [114] J. Hopcroft and R. Tarjan. Efficient planarity testing. *J. ACM*, 21(4):549–568, oct 1974.
- [115] J. E. Hopcroft and J. D. Ullman. *Introduction To Automata Theory, Languages, And Computation*. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edition, 1990.
- [116] S. Hougardy. Classes of perfect graphs. *Discrete Mathematics*, 306(19):2529–2571, 2006. Creation and Recreation: A Tribute to the Memory of Claude Berge.
- [117] S. il Oum and P. Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.
- [118] R. W. Irving and D. F. Manlove. The b -chromatic number of a graph. *Discrete Applied Mathematics*, 91(1):127–141, 1999.
- [119] L. Jaffke. *Bounded Width Graph Classes in Parameterized Algorithms*. PhD thesis, University of Bergen, 2020.
- [120] L. Jaffke, O. jounge Kwon, T. J. F. Strømme, and J. A. Telle. Mim-width III. Graph powers and generalized distance domination problems. *Theoretical Computer Science*, 796:216–236, 2019.
- [121] L. Jaffke, O. Kwon, T. J. F. Strømme, and J. A. Telle. Generalized Distance Domination Problems and Their Complexity on Graphs of Bounded mim-width. In C. Paul and M. Pilipczuk, editors, *13th International Symposium on Parameterized and Exact Computation (IPEC 2018)*, volume 115 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [122] L. Jaffke, P. T. Lima, and D. Lokshtanov. b -Coloring Parameterized by Clique-Width. In M. Bläser and B. Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, volume 187 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

- [123] L. Jaffke, P. T. Lima, and R. Sharma. b -coloring parameterized by pathwidth is xnlp-complete, 2022. arXiv e-prints arXiv:2209.07772 [cs.CC].
- [124] D. Y. Kang, O. joungh Kwon, T. J. F. Strømme, and J. A. Telle. A width parameter useful for chordal and co-comparability graphs. *Theoretical Computer Science*, 704:1–17, 2017.
- [125] R. M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972.
- [126] T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 1st edition, 1994.
- [127] D. König. *Theorie der endlichen und unendlichen Graphen: kombinatorische Topologie der Streckenkomplexe*. Mathematik und ihre Anwendungen 16. Akademische Verlagsgesellschaft m. b. h., 1936.
- [128] D. Král’ and R. Skrekovski. A theorem about the channel assignment problem. *SIAM Journal on Discrete Mathematics*, 16(3):426–437, 2003.
- [129] J. Kratochvíl, Z. Tuza, and M. Voigt. On the b -chromatic number of graphs. In G. Goos, J. Hartmanis, J. van Leeuwen, and L. Kučera, editors, *Graph-Theoretic Concepts in Computer Science*, pages 310–320, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.
- [130] P. K. Krause, L. Larisch, and F. Salfelder. The tree-width of c . *Discrete Applied Mathematics*, 278:136–152, 2020. Eighth Workshop on Graph Classes, Optimization, and Width Parameters.
- [131] V. R. Kulli and B. Janakiram. The maximal domination number of a graph. *Graph Theory Notes of New York*, 33:11–13, 01 1997.
- [132] C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51:45–64, 1962.
- [133] V. A. Leoni and E. G. Hinrichsen. $\{k\}$ -packing functions of graphs. In P. Fouilhoux, L. E. N. Gouveia, A. R. Mahjoub, and V. T. Paschos, editors, *Combinatorial Optimization*, pages 325–335, Cham, 2014. Springer International Publishing.
- [134] Z. Li, Z. Shao, F. Lang, X. Zhang, and J.-B. Liu. Computational complexity of outer-independent total and total roman domination numbers in trees. *IEEE Access*, 6:35544–35550, 2018.
- [135] N. Lindner and J. Reisch. An analysis of the parameterized complexity of periodic timetabling. *Journal of Scheduling*, 25:157–176, 2022.
- [136] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979.
- [137] H. Maimani, M. Momeni, S. Nazari Moghaddam, F. Rahimi Mahid, and S. Sheikholeslami. Independent double Roman domination in graphs. *Bulletin of the Iranian Mathematical Society*, 46:543–555, 2020.

- [138] M. Marzban and Q.-P. Gu. Computational study on a PTAS for planar dominating set problem. *Algorithms*, 6(1):43–59, 2013.
- [139] H. Matsuda, T. Ishihara, and A. Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science*, 210(2):305–325, 1999.
- [140] C. McDiarmid and B. Reed. Channel assignment on graphs of bounded treewidth. *Discrete Mathematics*, 273(1):183–192, 2003. EuroComb'01.
- [141] A. Meir and J. W. Moon. Relations between packing and covering numbers of a tree. *Pacific Journal of Mathematics*, 61(1):225–233, 1975.
- [142] Y. Mizutani and B. D. Sullivan. Parameterized Complexity of Maximum Happy Set and Densest k -Subgraph. In H. Dell and J. Nederlof, editors, *17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [143] M. Molloy and B. Reed. A bound on the strong chromatic index of a graph. *Journal of Combinatorial Theory, Series B*, 69(2):103–109, 1997.
- [144] A. Munaro and S. Yang. On algorithmic applications of sim-width and mim-width of (h_1, h_2) -free graphs. *Theoretical Computer Science*, 955:113825, 2023.
- [145] F. Nahani Pour, H. Abdollahzadeh Ahangar, M. Chellali, and S. M. Sheikholeslami. Global triple Roman dominating function. *Discrete Applied Mathematics*, 314:228–237, 2022.
- [146] M. Olsen. A general view on computing communities. *Mathematical Social Sciences*, 66(3):331–336, 2013.
- [147] Ø. Ore. *Theory of graphs*, volume XXXVIII of *Colloquium Publications*. American Mathematical Society, Providence, 3rd edition, 1962.
- [148] G. Pastukhov, A. Veremyev, V. Boginski, and O. A. Prokopyev. On maximum degree-based γ -quasi-clique problem: Complexity and exact approaches. *Networks*, 71(2):136–152, 2018.
- [149] J. Pattillo, A. Veremyev, S. Butenko, and V. Boginski. On the maximum quasi-clique problem. *Discrete Applied Mathematics*, 161(1):244–257, 2013.
- [150] A. Poureidi and N. J. Rad. On the algorithmic complexity of Roman $\{2\}$ -domination (Italian domination). *Iranian Journal of Science and Technology, Transactions A: Science*, 44:791–799, 2020.
- [151] N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.
- [152] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

- [153] P. Roushini Leely Pushpam and S. Padmapriya. Global Roman domination in graphs. *Discrete Applied Mathematics*, 200:176–185, 2016.
- [154] E. Sampathkumar and H. B. Walikar. The connected domination number of a graph. *Journal of Mathematical and Physical Sciences*, 13:607–613, 1979.
- [155] R. Sasák. Comparing 17 graph parameters. Master’s thesis, University of Bergen, 2010.
- [156] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994.
- [157] Z. Shao, J. Amjadi, S. M. Sheikholeslami, and M. Valinavaz. On the total double Roman domination. *IEEE Access*, 7:52035–52041, 2019.
- [158] Z. Shao, S. M. Sheikholeslami, S. Nazari-Moghaddam, and S. Wang. Global double Roman domination in graphs. *Journal of Discrete Mathematical Sciences and Cryptography*, 22(1):31–44, 2019.
- [159] C. Sloper. An eccentric coloring of trees. *Australasian Journal of Combinatorics*, 29:309–321, 2004.
- [160] L. J. Stockmeyer and V. V. Vazirani. NP-completeness of some generalizations of the maximum matching problem. *Information Processing Letters*, 15(1):14–19, 1982.
- [161] K. Sutner. Linear Cellular Automata and the Garden-of-Eden. *The Mathematical Intelligencer*, 11:49–53, 1989.
- [162] J. J. Sylvester. Chemistry and algebra. *Nature*, 17(284), 02 1878.
- [163] S. H. Sæther and M. Vatshelle. Hardness of computing width parameters based on branch decompositions over the vertex set. *Theoretical Computer Science*, 615:120–125, 2016.
- [164] J. A. Telle. *Vertex Partitioning Problems: Characterization, Complexity and Algorithms on Partial k -Trees*. PhD thesis, University of Oregon, 1994.
- [165] J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k -trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.
- [166] D. M. Thilikos, M. J. Serna, and H. L. Bodlaender. Constructive linear time algorithms for small cutwidth and carving-width. In G. Goos, J. Hartmanis, J. van Leeuwen, D. T. Lee, and S.-H. Teng, editors, *Algorithms and Computation*, pages 192–203, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.
- [167] M. Thorup. All structured programs have small tree width and good register allocation. *Information and Computation*, 142(2):159–181, 1998.
- [168] S. Tippenhauer. On planar 3-SAT and its variants. Master’s thesis, Freie Universität Berlin, 2016.

- [169] S. Vasiliev. *Algoritmos aproximados para problemas de optimización combinatoria*. PhD thesis, Universidad de Buenos Aires, 2018.
- [170] M. Vatshelle. *New width parameters of graphs*. PhD thesis, University of Bergen, 2012.
- [171] V. G. Vizing. Vertex colorings with given colors. *Diskret. Analiz*, 29:3–10, 1976.
- [172] L. Volkmann. On the signed total Roman domination and domatic numbers of graphs. *Discrete Applied Mathematics*, 214:179–186, 2016.
- [173] C.-X. Wang, Y. Yang, H.-J. Wang, and S.-J. Xu. Roman $\{k\}$ -domination in trees and complexity results for some classes of graphs. *Journal of Combinatorial Optimization*, 42:174–186, 2021.
- [174] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, 1980.
- [175] P. Zhang and A. Li. Algorithmic aspects of homophyly of networks. *Theoretical Computer Science*, 593:117–131, 2015.
- [176] X. Zhang, Z. Li, H. Jiang, and Z. Shao. Double roman domination in trees. *Information Processing Letters*, 134:31–34, 2018.