Optimization problems with propagation in graphs : Parameterized complexity and approximation

Morgan Chopin

Université Paris-Dauphine

N° attribué par la bibliothèque

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PARIS-DAUPHINE

Spécialité : **Informatique**

préparée au **LAMSADE**

dans le cadre de l'**École Doctorale de Dauphine**

présentée et soutenue publiquement par

Morgan CHOPIN

le 5 Juillet 2013

Problèmes d'optimisation avec propagation dans les graphes: Complexité paramétrée et approximation

Optimization problems with propagation in graphs: Parameterized complexity and approximation

sous la direction de Cristina Bazgan

Jury

Directeur de thèse :
Cristina Bazgan, professeur — LAMSADE, Université Paris-Dauphine

Rapporteurs :
Henning Fernau, professeur — Universität Trier
Ioan Todinca, professeur — LIFO, Université d'Orléans

Examinateurs :
Janka Chlebíková, senior lecturer — University of Portsmouth
Rolf Niedermeier, professeur — Technische Universität Berlin
Irena Rusu, professeur — LINA, Université de Nantes
Yann Vaxès, professeur — LIF, Aix-Marseille Université
Stéphane Vialette, directeur de recherche CNRS — LIGM, Université de Marne-la-Vallée

L'université n'entend donner aucune approbation ni improbation aux opinions émises dans les thèses : ces opinions doivent être considérées comme propres à leurs auteurs.

Contents

Remerciements

Trois ans et neuf mois. C'est exactement le temps qui s'est écoulé entre la signature de mon contrat doctoral et l'écriture de cette phrase. Quatre années de recherche qui ont constituées une expérience incroyablement enrichissante à la fois sur le plan scientifique mais également humain. Cependant, s'il y a bien une chose que j'ai apprise, c'est que la recherche ne se fait pas en solitaire et que de nombreuses personnes se cachent derrière chaque octet de ce fichier, devenu thèse. Je souhaite donc à présent leur dédier ce chapitre.

Je voudrais tout d'abord exprimer ma gratitude à mon directeur de thèse Cristina Bazgan. Merci de m'avoir fait confiance quatre ans plus tôt. Merci pour ta rigueur, ta franchise, tes conseils avisés, ton écoute et ta disponibilité qui ont su me guider tout au long de cette aventure.

Merci à Michael Fellows et Frances Rosamond pour leur inépuisable énergie à partager leurs connaissances. Je remercie en particulier Michael Fellows pour m'avoir introduit de manière avancée et avec dynamisme au domaine de la complexité paramétrée.

Merci à Rolf Niedermeier de m'avoir apporté son soutien puis accueilli pendant trois mois au sein de son équipe à Berlin. Ce séjour a largement contribué à approfondir mes connaissances du domaine et à donner de nouvelles perspectives à ma thèse. Je tiens également à remercier chaque membre de son équipe avec qui j'ai eu la chance de travailler: René van Bevern, Robert Bredereck, Jiehua Chen, Sepp Hartung, Falk Hüffner, Christian Komusiewicz, André Nichterlein, Manuel Sorge, Ondřej Suchý, et Mathias Weller.

Merci à Janka Chlebíková avec qui j'ai eu le plaisir de collaborer pendant trois mois à Portsmouth. Je tiens en particulier à la remercier non seulement d'avoir eu la gentillesse de relire des parties du présent manuscrit, mais également de m'avoir apporté son soutien lorsque je lui ai demandé.

Mes remerciements vont également à Bernard Ries, Florian Sikora, Zsolt Tuza, et Daniel Vanderpooten avec qui j'ai eu la bonne fortune de collaborer à un certain moment et qui ont contribués également à forger mon esprit scientifique.

Je souhaite aussi exprimer ma gratitude à Pierre-Henri Wuillemin qui m'a toujours apporté son aide lorsque je la lui ai demandée.

Merci à Ioan Todinca d'avoir accepté d'être dans mon jury de présoutenance et dont les commentaires ont permis d'améliorer le présent document.

Ces remerciements seraient incomplets si je n'en adressais pas à l'ensemble de l'équipe du LAMSADE pour leur amitié et leur support. Je tiens à remercier plus particulièrement mes colocs de bureau pour les nombreux bons moments passés ensemble: Pierre-Emmanuel Arduin, Amal Benhamiche, Basile Couëtoux, Edouard Bonnet, Yann Dujardin, Florian Jamain, Sébastien Martin, Renaud Lacour, Amine Louati, Lydia Tlilane, Abdallah Saffidine, Raouia Taktak, et Emeric Tourniaire.

Je voudrai remercier tout spécialement Sonia Toubaline pour son appui ininterrompu allant de mes premiers jours de thèsard un peu perdu jusqu'aux derniers jours d'écriture en relisant certaines parties du présent manuscrit. Merci de ta constante gentillesse, de ton amitié et, surtout, passe le bonjour au "précieux" de ma part.

Merci à tous mes amis Chti qui m'ont permis de sortir de ma "bulle" le temps d'une soirée, d'un week-end ou plus. Merci également à mes deux (trois?!) bretons préférés de m'emmener chaque année m'aérer la tête à la montagne.

Merci à mon frère Julien dont les quatre ans d'avance sur moi dans la voie de la recherche m'ont toujours guidés au fil de mes études. Thanks bro!

Merci à Céline Zajakala qui fait mon bonheur au quotidien. Sans toujours le savoir, tu as été un véritable manager durant ces années, à me ramener sur terre quand il le fallait, à supporter les moments difficiles et autres innombrables "attend, je dois terminer un truc pour demain". Je ne crois pas pouvoir trouver les mots pour dire à quel point ta bienveillance et ta bonne humeur à toute épreuve ont été et sont essentielles pour moi.

Enfin, mon dernier remerciement s'adresse à ma mère qui a toujours su prendre les bonnes décisions pour moi et m'a permis de faire les études qui me plaisent. Sans cela, les pages qui suivent n'auraient jamais pu voir le jour (et aussi parce que je me suis toujours assis "à côté de quelqu'un qui a une très bonne moyenne")

Résumé de la thèse

Depuis ces dix dernières années, l'évolution des moyens de communication ainsi que la démocratisation d'Internet ont fait des réseaux sociaux un enjeu crucial à la fois du point de vue social bien sûr, mais également économique [52]. En particulier, la *propagation (ou diffusion) d'information* au sein de telles structures n'a cessé de recevoir un interêt croissant de la part de la commmunauté scientifique, motivé par des applications telles que le marketing viral, la diffusion de rumeur ou même la santé publique. Il existe en effet un lien étroit entre la topologie d'un réseau social et la propagation d'une maladie au sein d'une population [41]. Le terme "information" est donc à prendre au sens large puisqu'il peut faire référence à une rumeur, une maladie, un feu, un message publicitaire, etc. Par "propagation", nous sous-entendons un mécanisme qui définit la manière dont l'information se transmet d'un individu à l'autre à travers tout le réseau. Afin de pouvoir étudier formellement ces phénomènes de diffusion, plusieurs modèles théoriques basés sur la théorie des graphes ont été proposés [47, 97, 29, 72, 36, 37, 70, 81, 80]. En effet, étant donné que les réseaux sociaux consistent en un ensemble d'individus inter-connectés selon une relation pré-déterminée (relation d'amitié, de travail, amoureuse, etc.), il est naturel de représenter ces derniers à l'aide d'un graphe (voir Figure 1). On dit alors qu'un sommet du graphe est dans l'état "activé" si l'information lui a été transmise. Selon le contexte, le terme "activé" peut correspondre à un individu infecté par un virus, ou encore à une personne ayant connaissance d'une rumeur.

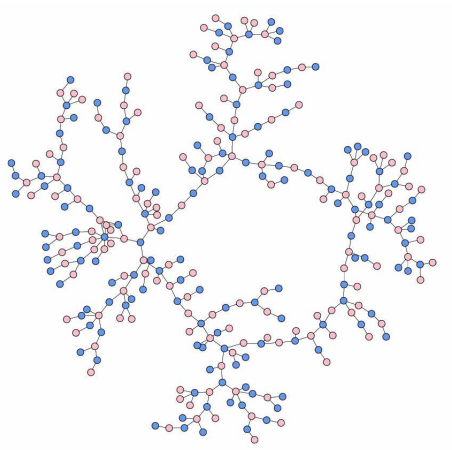


Figure 1: L'aspect "arborescent" d'un graphe représentant des relations amoureuses dans une école Américaine (image créée par Mark Newman d'après Bearman et al. [16]). Les sommets en bleus (gris sombre) et roses (gris clair) correspondent, respectivement, aux garçons et filles. Il y a une arête entre deux individus s'ils ont vécu une relation amoureuse au cours des 18 mois de l'étude.

Une fois qu'un tel graphe a été établi, la prochaine étape consiste à en extraire des informations pertinentes. À ce titre, plusieurs études se sont interessées, entre autre, aux questions suivantes: Comment déterminer les individus les plus influents? Quelle partie de la population est la plus encline à résister à une épidémie? Quelle est la meilleur

stratégie de vaccination possible si une maladie vient à se propager? Chacune de ces interrogations peut être abordée comme un problème algorithmique, et tout l'enjeu est alors de savoir si "calculer" les réponses correspondantes est faisable en pratique. Cette thèse se donne donc pour but d'examiner la complexité de résolution de ces problèmes du point de vue algorithmique. Pour ce faire, nous nous proposons d'étudier la complexité exacte et paramétrée ainsi que l'approximation de problèmes d'optimisation combinatoire impliquant un processus de diffusion dans les graphes. Ce travail est structuré autour de trois problématiques décrites dans les paragraphes suivants. Pour chacune d'entre elles, nous définissons le ou les problèmes considérés, nous en faisons l'état de l'art et présentons les résultats obtenus dans ce travail.

Maximiser la diffusion d'information (Chapitre 3).

Maximiser la diffusion d'information dans un réseau social est un problème intervenant dans des contextes très divers comme, par exemple, le viral marketing. Cette technique commerciale consiste à promouvoir un produit auprès de personnes influentes à travers un message persuasif. L'objectif est alors de créer un effet de "bouche à oreille" pour que ce message se diffuse le plus largement possible. L'originalité de cette approche vient du fait que ce sont les clients eux-même qui font la publicité du produit. Le problème d'optimisation qui en découle naturellement consiste en la donnée d'un graphe, une valeur de seuil $\text{thr}(v)$ associée à chaque sommet v de ce graphe, ainsi que la règle de propagation suivante: un sommet devient actif s'il possède au moins $\text{thr}(v)$ voisins activés. Le processus de propagation se déroule alors en plusieurs étapes et se termine lorsque qu'aucun nouveau sommet ne peut être activé. Étant donné ce modèle de diffusion, l'objectif est alors de trouver et activer un ensemble de sommets de taille minimum de telle sorte que tous les sommets du graphe soient activés à la fin du processus de propagation. Ce problème est connu dans la littérature sous le nom de *target set selection*[1] et a été introduit par Chen [35].

Ce problème a été montré NP-hard même dans les graphes bipartis de degré borné avec des seuils au plus égaux à deux [35]. Il est également difficile à approximer à un ratio $O(2^{\log^{1-\varepsilon} n})$ pour tout $\varepsilon > 0$ même pour des graphes de degré borné avec des seuils d'au plus deux [35]. Ce rapport d'inapproximation reste valide pour des seuils majoritaires (*i.e.* le seuil de chaque sommet est égal à son degré divisé par deux). Dans le cas des seuils unanimes (*i.e.* le seuil de chaque sommet est égal à son degré), le problème correspond exactement au problème *vertex cover*. Par conséquent, il est 2-approximable en temps polynomial et est difficile à approximer à un ratio mieux que 1.36 [46]. Concernant sa compléxité paramétrée, le problème est W[2]-hard pour le paramètre "taille de la solution", même dans les graphes bipartis avec des seuils majoritaires ou au plus égaux à deux [90]. Cependant, Nichterlein et al. [90] ont donnés plusieurs algorithmes paramétrés lorsque les paramètres sont liés à la structure du graphe. De plus, Ben-Zwi et al. [17] ont montré que le problème est résoluble en temps polynomial pour des graphes de largeur arborescente bornée.

À la lumière de ce dernier constat, nous proposons, dans un premier temps, d'explorer plus en avant la complexité paramétrée de *target set selection* à l'aide de paramètres structurels. Nos résultats sont résumés dans la figure 2. Une des conclusions de cette étude est que supposer des seuils constants permet de rendre le problème traitable en pratique. A priori, on pourrait penser que c'est une hypothèse assez restrictive, mais dans de nombreux contextes applicatifs, tel que le viral marketing, supposer de tels seuils

[1] Nous gardons la dénomination anglaise pour le nom des problèmes.

est suffisant. En effet, indépendamment du nombre de mes amis, il pourrait suffir que cinq d'entre eux achètent un certain produit pour que je sois convaincu de son utilité et l'achète à mon tour. Dans un second temps, nous avons envisagé l'approximation paramètrée du problème complémentaire *max influence*: Étant donné un graphe, une valeur de seuil $\text{thr}(v)$ associée à chaque sommet v de ce graphe, et un entier $k > 0$, l'objectif est de trouver et activer un ensemble de sommets de taille au plus k de telle sorte que le nombre de sommets activés à la fin du processus de propagation soit maximum. Le processus de propagation est le même que celui défini pour *target set selection*. Ce problème est en fait la version déterministe du problème introduit par Kempe et al. [72]. Nos résultats sont résumés dans le tableau 1. Notons qu'il existe deux façons de mesurer la taille de la solution en comptant ou non l'ensemble initialement activé. De ce fait, nous définissons, respectivement, les deux problèmes *max closed influence* et *max open influence*. On rappelle qu'un problème d'optimisation est *$\alpha(n)$-approximable en temps fpt par rapport au paramètre k* si le problème est $\alpha(n)$-approximable en temps $f(k)\cdot n^{O(1)}$ où f est une fonction qui ne dépend que de k. Il est intéressant de noter que *max open influence*, pour le cas unanime, est $\alpha(n)$-approximable en temps fpt par rapport au paramètre k pour toute fonction croissante α (Corollary 32) alors qu'il est W[1]-hard par rapport à ce même paramètre (Theorem 29) et inapproximable en temps polynomial à un ratio $n^{1-\varepsilon}$ pour tout $\varepsilon > 0$ si NP $\neq$ ZPP (Theorem 30).

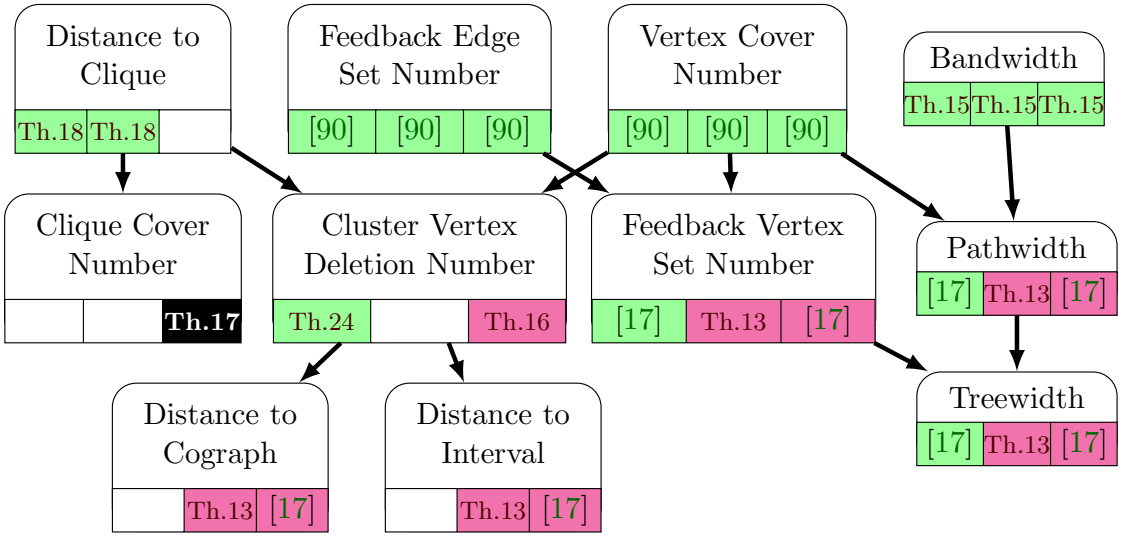


Figure 2: Panorama de nos résultats de complexité paramètrée pour *target set selection* avec des paramètres structurels. Les trois cases en-dessous de chaque paramètre indiquent un résultat pour des seuils (de gauche à droite) bornés par une constante, majoritaire, et sans restriction. Une arête joignant un paramètre k_2 à un autre paramètre k_1 en dessous de k_2 implique l'existence d'une constante $c > 0$ telle que $k_1 \leq c \cdot k_2$. Le paramètre "Distance to $\mathcal{C}$" où $\mathcal{C}$ est une classe de graphe correspond au nombre minimum de sommets à retirer de sorte que le graphe obtenu appartient à $\mathcal{C}$. Pour le paramètre "clique cover number", la case noire indique que le problème est NP-hard pour une valeur constante du paramètre. La couleur violette (gris sombre) indique un résultat de W[1]-hardness alors que la couleur verte (gris clair) signifie que le problème admet un algorithme paramétré. Lorsque la case est blanche, la question est ouverte.

Les résultats présentés dans cette section sont basés sur les papiers suivants:

		max open influence		***max closed influence***	
Seuils	Bornes	temps poly.	temps fpt	temps poly.	temps fpt
Cons.	Sup. Inf.	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$ (Th.27)
Maj.	Sup. Inf.	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$ (Th.26)
Una.	Sup. Inf.	2^k (Th.31) $n^{1-\varepsilon}, \forall \varepsilon > 0$ (Th.30)	$\alpha(n), \forall \alpha$ (Th.32) ?	2^k $1+\varepsilon$ (Th.36)	$\alpha(n), \forall r$?

Table 1: Tableau regroupant nos résultats d'approximation pour *max open influence* et *max closed influence* avec des seuils constants (Cons.), majoritaires (Maj.) et unanimes (Una.). Les résulats d'approximation paramétrés sont établis par rapport au paramètre k pour les deux problèmes. La valeur n correspond à la taille du graphe.

▶ M. Chopin, A. Nichterlein, R. Niedermeier, and M. Weller, *Constant Thresholds Can Make Target Set Selection Tractable*, Proceedings of the 1st Mediterranean Conference on Algorithms (MedALG 2012), LNCS 7659, pp. 120–133, 2012.

▶ C. Bazgan, M. Chopin, A. Nichterlein and F. Sikora, *Parameterized Approximability of Maximizing the Spread of Influence in Networks*, Proceedings of the 19th Annual International Computing and Combinatorics Conference (COCOON 2013), LNCS 7936, pp. 543–554, 2013.

Déterminer un ensemble inoffensif (Chapitre 4).

Dans l'étude précédente, nous avons souligné la difficulté de traitabilité du problème *target set selection* et avons donc proposé d'autres approches afin d'obtenir des résultats positifs. Au regard de la nature intraitable de ce problème, il est intéressant de se poser la question de la complexité du problème inverse que l'on nomme *harmless set*: Étant donné un graphe, une valeur de seuil $\text{thr}(v)$ associée à chaque sommet v, et un entier $k > 0$, l'objectif est de trouver un ensemble de sommets de taille au moins k de sorte qu'activer des sommets quelconque dans cet ensemble ne permet d'activer aucun sommet par propagation. Autrement, on souhaite trouver un ensemble S de sommets tel que **tout** sommet v du graphe a un nombre de voisins dans S inférieur à $\text{thr}(v)$. On appelle alors l'ensemble S un ensemble *inoffensif*. Ce problème pourrait intervenir, par exemple, dans un contexte épidémique. L'objectif serait alors de déterminer la résistance naturelle d'une population face à la propagation d'un virus. Il est intéressant de noter que ce problème peut être mis en relation avec le problème (σ, ρ)-*dominating set* introduit par Telle [100]: Étant donné un graphe $G = (V, E)$, deux ensembles d'entiers non-négatifs σ et ρ, et un entier $k \geq 1$, l'objectif consiste à trouver un ensemble $S \subseteq V$ de taille au plus k tel que pour tout sommet $v \in V$, le nombre de voisins de v dans S appartient à σ et le nombre de voisins de v n'appartenant pas à S appartient à ρ. On dit alors que S est un ensemble (σ, ρ)-dominant. Il se trouve que si tous les seuils sont égaux alors *harmless set* est équivalent à (σ, ρ)-*dominating set of size* k [61] (qui demande un ensemble (σ, ρ)-dominant de taille exactement k) avec $\sigma = \rho = \{0, \ldots, \text{thr}_{\max}\}$ où $\text{thr}_{\max}$ est la valeur du seuil maximum. Puisque ce dernier problème est dans W[1] [61], cela implique que *harmless set* est également dans W[1] si tous les seuils sont égaux. À notre connaissance, c'est le seul résultat qui peut être transféré à *harmless set*.

Dans cette thèse, nous avons étudié la complexité paramétrée (voir le tableau 2) ainsi que l'approximation du problème de maximisation associé *max harmless set*. Nous avons établi que *max harmless set* est inapproximable à un ratio $n^{1-\varepsilon}$ pour tout $\varepsilon > 0$ avec des seuils d'au plus deux. Si les seuils sont unanimes, le problème est APX-hard et est 3-approximable en temps linéaire. Nous donnons également un schéma d'approximation polynomiale pour les graphes planaires.

Seuils	***harmless set***	***dual harmless set***
General	W[2]-complete (Th.42)	W[2]-hard
Constants	W[1]-complete (Th.44)	FPT (Th.48)
Majoritaires	W[1]-hard (Th.44)	FPT (Th.48)
Unanimes	FPT (Th.46)	W[2]-hard*

Table 2: Nos résultats de complexité paramétrée pour *harmless set* et son dual *dual harmless set* où l'objectif est de trouver un ensemble inoffensif de taille au moins $n-k$ où n est la taille du graphe. Le paramètre est k pour les deux problèmes. Le résultat marqué par * est dû à l'équivalence entre *dual harmless set* et le problème *total dominating set* qui a été montré W[2]-hard [61]

Les résultats présentés dans cette section sont basés sur le papier suivant:

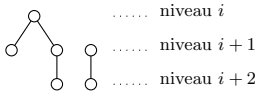
▶ C. Bazgan and M. Chopin, *The robust set problem: parameterized complexity and approximation*, Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS 2012), LNCS 7464, pp. 136–147, 2012.

Contenir la propagation d'information (Chapitre 5).

Dans cette partie finale, nous avons étudié le problème *firefighter*. De manière informel, l'objectif est ici de stopper la propagation d'information au sein d'un réseau en ayant la possibilité de "protéger" certains sommets. Un sommet protégé ne pouvant plus être activé. Ce problème a été initialement introduit par Hartnell en 1995 [64], et a depuis été très largement étudié [7, 27, 31, 55, 89, 54]. Il est défini de la manière suivante: initialement un sommet particulier d'un graphe est activé. À chaque pas de temps, on applique successivement les deux étapes suivantes: 1) Protéger un sommet non-activé du graphe; 2) Tous les sommets non-protégés et adjacents à un sommet activé sont activés. Le processus se termine lorsque plus aucun nouveau sommet ne peut être activé. Un sommet est alors considéré comme sauvé s'il n'est pas activé. Étant donné un entier $k > 0$, l'objectif est de trouver une stratégie de protection des sommets permettant de sauver au moins k sommets.

Ce problème a été montré NP-difficile dans les graphes bipartis [82], graphes cubiques [75], et graphes de disque-unité [58]. Finbow et al. [55] ont montré que le problème est NP-hard même dans les arbres de degré au plus trois et résoluble en temps polynomial dans les graphes de degré au plus trois si le sommet initialement activé est de dégré au plus deux. De plus, le problème *firefighter* est résoluble en temps polynomial dans les caterpillars et les P-arbres [82].[2] Du point de vue de l'approximation, la version

[2]Un P-arbre [82] est un arbre qui ne contient pas la configuration suivante:



maximisation du problème, qui consiste à maximiser le nombre de sommets sauvés, est $\frac{e}{e-1}$-approximable dans les arbres [27] et non $n^{1-\varepsilon}$-approximable dans les graphes généraux pour tout $\varepsilon > 0$, si $\mathsf{P} \neq \mathsf{NP}$ [7]. Pour les arbres où chaque sommet a au plus trois fils, le problème est 1.3997-approximable [69]. Peu de résultats sont connus concernant la complexité paramétrée du problème. Cai et al. [27] fournissent des algorithmes paramétrés dans le cas des arbres pour chacun des paramètres suivants: le nombre de sommets sauvés, le nombre de sommets brûlés, et le nombre total de sommets protégés. Pour le paramètre "nombre de sommets sauvés", les auteurs donnent un noyau polynomial.

Dans cette thèse, nous considérons la version plus générale du problème où $b \geq 1$ (appelé budget) sommets peuvent être protégés à chaque pas de temps. Nous étudions également le dual noté *dual firefighter* dont l'objectif est de sauver au moins $n - k_b$ sommets où n est la taille du graphe et k_b est un entier positif. Pour terminer, nous considérons le problème *bounded firefighter* qui est défini de manière similaire au problème *firefighter* excepté que l'on est autorisé à protéger un total d'au plus $k_p \geq 1$ sommets, où k_p est un entier donné en entrée du problème. Nous montrons que le problème *firefighter* est NP-complet dans les arbres de degré au plus $b + 3$ ainsi que dans les arbres de *pathwidth* au plus trois. Cependant, nous montrons que le problème est résoluble en temps polynomial pour la classe des graphes dont le degré maximum et le *pathwidth* sont bornés. Nous fournissons également un algorithme polynomial pour résoudre le problème (et la version pondérée correspondante) pour une sous classe d'arbres de *pathwidth* deux, les k-caterpillars. Nous établissons des bornes de compléxité paramétrée inférieures et supérieures pour les problèmes *firefighter*, *dual firefighter*, et *bounded firefighter* dans les graphes généraux par rapport aux paramètres standards (see Figure 3). Nous donnons également des algorithmes paramétrés dans les arbres qui améliorent les résultats obtenus par Cai et al. [27]. Nous répondons également à plusieurs questions ouvertes de [27]. De plus, nous établissons plusieurs algorithmes paramétrés par rapport à des paramètres liés à la structure du graphe (see Figure 4). Pour terminer, nous observons que la version minimisation du problème *firefighter* est inapproximable à un ratio $n^{1-\varepsilon}$ même dans les arbres pour tout $\varepsilon > 0$ et tout budget $b \geq 1$ si $\mathsf{P} \neq \mathsf{NP}$. Nous répondons de manière négative à une question ouverte de Finbow and MacGillivray [54].

	firefighter k	***bounded firefighter*** k_p	***dual firefighter*** k_b
	W[1]-hard	W[1]-hard	W[1]-hard
Noyau poly. ?	no	no	no
Budget	W[1]-hard	W[1]-hard	FPT
Noyau poly. ?	no	no	no
Treewidth	FPT	FPT	?
Noyau poly. ?	?	no	no

Figure 3: Résumé de nos résultats de compléxité paramétrée incluant la paramétrisation standard. À chaque colonne est associé un problème et son paramètre standard. À chaque ligne, excepté la première, correspond également un paramètre. L'intersection d'une ligne et d'une colonne donne un résultat de complexité paramétrée par rapport au paramètre combiné de la ligne et de la colonne.

Les résultats présentés dans cette section sont basés sur les papiers suivants:

▶ C. Bazgan, M. Chopin and M. R. Fellows, *Parameterized complexity of the firefighter problem*, Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011), LNCS 7074, pp. 643–652, 2011.

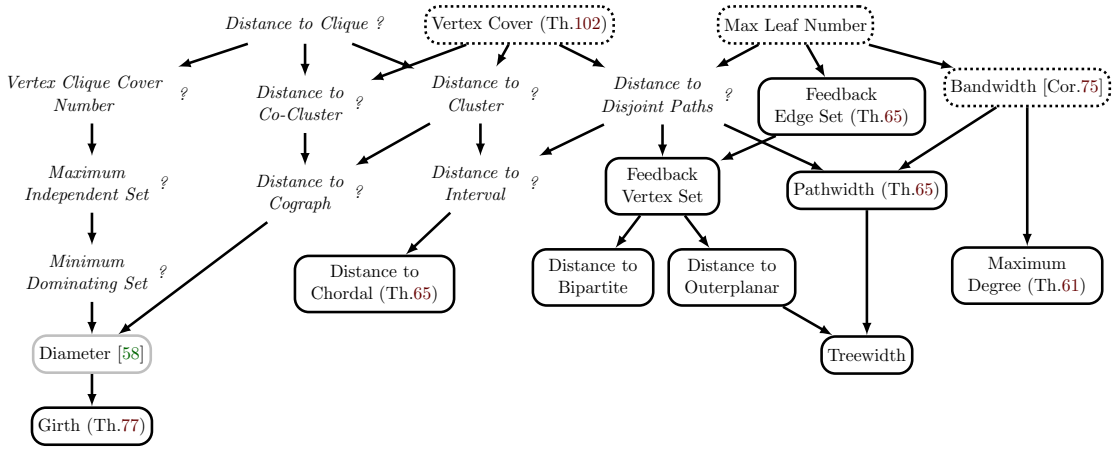


Figure 4: Nos résultats de complexité paramétrée pour le problème *firefighter* par rapport à des paramètres structurels. Un rectangle en pointillé indique que le problème admet un algorithme paramétré pour ce paramètre, un rectangle gris montre un résultat de W[1]-hardness, et un rectangle avec des bords noirs indique le problème est NP-hard pour une valeur constante de ce paramètre. Un rectangle avec un "?" indique une question ouverte. Pour le paramètre "diameter", Fomin et al. [58] ont montré que *firefighter* est dans XP.

▶ C. Bazgan, M. Chopin and B. Ries, *The firefighter problem with more than one firefighter on trees*, Discrete Applied Mathematics 161(7-8), 899-908, 2013.

▶ C. Bazgan, M. Chopin, M. R. Fellows, F. V. Fomin, E. J. van Leeuwen and M. Cygan, *Parameterized Complexity of Firefighting*, submitted.

▶ J. Chlebíková and M. Chopin, *The Firefighter Problem: A Structural Analysis*, ongoing paper.

Chapter

1 Introduction

Over the last past decade, the evolution of communication technology together with the democratization of Internet have made *social networks* a major economic and social stake [52]. In particular, the *propagation (or diffusion) of information* through these networks has gained a lot of interest driven by applications such as viral marketing, rumor spreading or even public health (for instance, there is a close connection between the topology of social networks and the propagation of a disease through a population [41]). The term "propagation of information" should be considered in a broad sense here as it may appear in various contexts. In order to better understand its meaning, let us consider the following real-world examples. In 1996, the email service *Hotmail* added the following simple message to the footer of every mail sent out by users "`Get your free email at Hotmail`". As a result, the number of subscribers grew by 12 million in 18 months. In this case, the propagated information was a textual advertisement message that goes (propagates) from one user to many others. As a matter of fact, this marketing technique is called "viral advertisement" due to its analogy with the spread of viruses or computer viruses. Another example is from a study of Christakis and Fowler [40]. In their work, the authors analyzed the obesity propagation during 32 years through a social network having more than 12000 persons. They found that having obese friends increase by 57% the chance of developing obesity. Moreover, they also observed that obesity may influence up to three degrees of separation *i.e.* if the friend of a friend of my friend is obese then my obesity risk is increased. Here the propagation is clearly of psychological nature. To formally capture these diffusion phenomenons, several theoretical models have emerged recently [47, 97, 29, 72, 36, 37, 70, 81, 80]. As social networks consist of individuals who are linked together by a pre-determined relationship, they are often represented as graphs. A graph is a mathematical object that comes up with a collection of *vertices* together with a collection of *edges* that connect pairs of vertices (see Figure 1.1).

Given a graph that represents a social network, a natural next step is to determine what kind of knowledge one can learn from it. For instance, several studies investigate the following questions among others: who are the most influencer individuals? what part of a population is the most resistant to an epidemic outbreak? in the case of a spread of a disease, what is the best vaccination strategy? Each of these interrogations can be regarded as a particular problem to solve, and the question that naturally follows is whether it is an easy task or not. This leads us to the main objective of this thesis which is to address the complexity of these problems from the *computer science* point of view. More specifically, the goal is to design efficient *algorithms* that compute the answer of the previous questions. By "efficient" we mean that the running time of these algorithms on a computer is required to be reasonable *i.e.* they should not run more than a few days. If such an algorithm exists, we say that the problem is *practically solvable*. As a matter of fact, all problems in this work are *combinatorial* problems. At its most general form, a combinatorial problem is a problem for which the goal is to find an optimal solution among a large finite set of solutions. Consider for example a well known combinatorial problem "Traveling Salesman Problem (TSP)" [8]: Given a list of cities and distances between each pair of cities, TSP consists of finding a shortest possible route that visits each city exactly once and returns to the origin city. One might think that it suffices to exhaustively

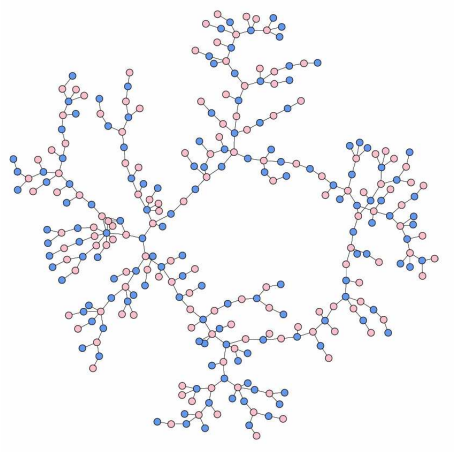


Figure 1.1: The "tree-like" aspect of a graph representing romantic relationships in an American high school (image drawn by Mark Newman from Bearman et al. [16]). Blue (dark gray) and pink (light gray) vertices correspond to male and female, respectively. An edge joins two individuals if they were romantically involved during the 18 months in which the study was conducted.

check every route to eventually find the optimal one. However, a typical combinatorial problem has a number of possible solutions which is so large that the best solution could not be obtain within a reasonable amount of time using this naive technique. This is what we call the *combinatorial explosion*. Considering the traveling salesman problem with only 20 cities, such exhaustive algorithm may take centuries before we get the optimal solution, even with the world-fastest computer. To be convinced, let us consider a really fast computer able to compare two possible routes and say which one is the shortest in less than 0.000000001 second or, equivalently, one nanosecond *i.e.* faster than the time taken for light to travel one meter. It turns out that for 20 cities, the number of possible routes is about 2.4×10^{18}. Since the computer needs to compare every route with each other, this would take 2.4×10^{18} nanoseconds to find the shortest route which corresponds roughly to 770 years! This is definitely not a reasonable running time to solve the problem. The central question is then to determine whether there exists a better approach *i.e.* for a given problem, is there an algorithm that could possibly solve it in reasonable time? This leads us to the concept of computational "hardness" of a problem. This notion is rigorously defined in *computational complexity theory* but, for the sake of clarity, we only say here that a problem is hard if its solution requires significant resources (essentially time and memory) to be computed. Computer scientists have grouped problems into classes based on how long they take to be solved. The class NP gathers problems for which an answer can be verified in a reasonable amount of time. Some problems of NP can in fact be solved quickly. Those problems are said to be in P, which stands for polynomial time. However, there are other problems in NP which have never been solved in polynomial time and widely believed of not being so. Those latter problems are said NP-hard (like the traveling salesman problem). In fact, problems in NP and P are required to be *decision problems*, *i.e.* a problem with only two possible solutions "yes" or "no". However, this is not restrictive since every combinatorial optimization problem has a computationally equivalent decision problem. For example, computing an optimal solution for the traveling salesman problem is at least as hard as to solve its decision version: given an integer k, is there a route of length at most k that visits each city exactly once and returns to the

origin city?

In this thesis, we consider combinatorial optimization problems related to the diffusion of information in social networks. We then classify them either by proving that they are polynomial-time solvable or by proving that they are in fact NP-hard. In the last case, we try to rely on other type of methods such as approximation or parameterized algorithms to solve them. For an approximation algorithm, the basic idea is to release the constraint of optimality while the running time is required to be reasonable, and the computed solution must be guaranteed to be "close" to the optimum. A parameterized algorithm tries to get rid of the easy-to-solve parts of a problem so that it only remains what makes the problem difficult. The idea is to cope with a smaller equivalent version of the problem, thus making algorithms running faster on it. We can even combine the two approaches to get a parameterized approximation algorithm.

Let us consider a viral marketing problem as an illustration of the problems studied in this thesis. The aim of viral marketing is to advertise a product to the most influential customers who are most likely to produce a "word-of-mouth" effect through their social network. The advantage of this technique is that the customers perform themselves the advertisement of a product, saving thus a lot of money for the company. The Figure 1.2 depicts the partial social network of a customer called "Bob" by the use of an undirected graph. Each vertex corresponds to a customer and an edge between two vertices indicates that these two individuals know each other. The number inside each vertex is called "threshold". A customer with threshold t is convinced of a product's usefulness and buys it if at least t of its friends have one. In this thesis, we speak in a more general sense and thus say that a vertex of a graph is "active" instead of "convinced". Depending of the application context, the term "active" could also mean "infected" or "burned" as well. In this example, Bob would buy an xphone if three of his friends have it. Suppose now that the company can offer a very limited number of xphone, say two in this example. The question is now to find who are the two customers to give the product. Of course, the stake here is to choose the two most influencer ones. In fact, one optimal solution would be to choose Edward and Carol. In this little example, the optimal solution is very easy to find and an exhaustive search approach would have solved the problem quickly. However, real-world social networks might involve over millions of individuals with complex relationship, thus making exhaustive algorithms useless.

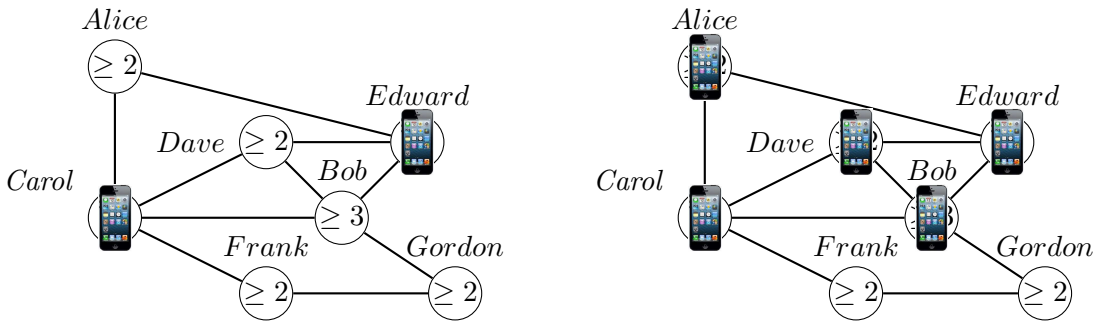


Figure 1.2: Initially, Edward and Carol are offered an xphone. Then, by a "word-of-mouth" process, Alice and Dave are convinced (or "activated") and buy an xphone. Since Carol, Edward and now Dave have the mobile phone, Bob decides to get one.

This thesis is structured around three different problematics described hereafter and further developed in the subsequent chapters. Firstly, we recall basic theoretical background in Chapter 2. In Chapter 3, we consider two problems with the objective of max-

imizing the spread of influence in social networks. The first one is the target set selection problem. Given a social network represented as a graph and a threshold value $\text{thr}(v)$ associated to each vertex v, the task is to find and activate a vertex subset of minimum size such that **all** the vertices become active at the end of the propagation process defined as follows. A vertex v becomes active if at least $\text{thr}(v)$ of its neighbors are active. The propagation process proceeds in several steps and stops when no further vertex becomes active. The next investigated problem is maximum influence which is defined similarly except that the input has an extra integer k (it corresponds to some "budget") and we ask to activate at most k vertices such that the total number of activated vertices at the end of the propagation process is maximized (this is exactly the problem in the above viral marketing example). In Chapter 4, we turn our attention to a converse objective: Find the largest set of vertices such that if any vertices get activated in it then no new vertex can be activated by the application of the propagation rule. One motivation for this problem arises from the context of preventing the spread of dangerous ideas or epidemics. It also corresponds to the objective of finding a population resistant to an epidemic outbreak. In Chapter 5, we now consider the problem of containing a malicious agent (fire, virus, ...) which has already started to propagate through a network. Initially, the outbreak starts at a single vertex of a graph. At each time step, we have to choose one vertex which will be protected. Then the outbreak spreads to all unprotected neighbors of the "infected" vertices. The process ends when the outbreak can no longer spread, and then all vertices that are not infected are considered as saved. The objective consists of choosing, at each time step, a vertex which will be protected such that a maximum number of vertices in the graph is saved at the end of the process. Finally, the Chapter 6 provides conclusions and future research directions.

Chapter 2

Preliminaries

THE purpose of this chapter is to give the basic backgrounds on classical and parameterized complexity, approximation as well as some basic graph theory concepts and notations used throughout this thesis. For more details about parameterized complexity theory, the reader is referred to the books of Downey and Fellows [49], Niedermeier [91], and Flum and Grohe [57]. Concerning the approximation theory, we recommend the following books of Ausiello, Crescenzi, Gambosi, Kann, Marchetti-Spaccamela, and Protasi [12], Hochbaum [67], Vazirani [101], and Williamson and Shmoys [103].

2.1 Graph theory: terminology and notations

We denote a graph by an ordered pair $G = (V, E)$ where V is the set of vertices and $E \subseteq V \times V$ the set of edges. We now define basic graph terminology.

A vertex $v \in V$ is *adjacent* to another vertex $u \in V$ if there is an edge $uv \in E$ connecting them. The *neighbors* (or *neighborhood*) of a vertex v is the set of vertices adjacent to v. The *degree* of a vertex is the number of its neighbors.

Two vertices are said *twins* if they have the same neighborhood. They are called *true twins* if they are moreover neighbors, *false twins* otherwise.

A *path* is either a single vertex or a graph where every vertex has degree one or two, and exactly two vertices have degree one (called *endpoints*). The length of a path is the number of edges.

The *diameter* of a graph is the longest shortest path between any two vertices.

A graph is *connected* if there is a path joining every pair of vertices.

A *cycle* is a connected graph where every vertex has degree two.

A *subgraph* $H = (V', E')$ of G is a graph where $V' \subseteq V$ and $E' \subseteq E$.

A subgraph $H = (V', E')$ of G is said to be *induced* by V' if, for any pair of vertices $u, v \in V'$, we have $uv \in E'$ if and only if $uv \in E$.

A *linear layout* of G is a bijection $\pi : V \to \{1, \ldots, n\}$. For convenience, we express π by the list $L = (v_1, \ldots, v_n)$ where $\pi(v_i) = i$. Given a linear layout L, we denote the distance between two vertices in L by $d_L(v_i, v_j) = |i - j|$.

The *cutwidth* $\mathrm{cw}(G)$ of G is the minimum integer k such that the vertices of G can be arranged in a linear layout $L = (v_1, \ldots, v_n)$ in such a way that, for every $i = 1, \ldots, n - 1$,

there are at most k edges with one endpoint in $\{v_1, \ldots, v_i\}$ and the other in $\{v_{i+1}, \ldots, v_n\}$

The *bandwidth* $\mathrm{bw}(G)$ of G is the minimum integer k such that the vertices of G can be arranged in a linear layout $L = (v_1, \ldots, v_n)$ in such a way that, for every edge v_iv_j of G we have $d_L(v_i, v_j) \leq k$.

If the graph G is a *directed graph* (or *digraph* for short), then we need to adjust some previously established terminologies. In a directed graph every edge uv is called an *arc* and is directed either from u to v, denoted (u, v), or from v to u, denoted (v, u). We will denote by A rather than by E the set of arcs in G. The *in-neighbors* (resp. *out-neighbors*) of a vertex v is the set of vertices $N^+(v) = \{u \in V : (u, v) \in A\}$ (resp. $N^-(v) = \{u \in V : (v, u) \in A\}$).

Unless otherwise specified, all graphs in this thesis are **undirected**, **finite** (bounded number of vertices) and **simple** (at most one edge connects two vertices).

Notations. Let $G = (V, E)$ be a graph, we will use the following standard notations.

- $N_G(v)$: open neighborhood of a vertex v *i.e.* $N_Gv) = \{u \in V : uv \in E\}$.
- $N_G[v]$: close neighborhood of a vertex v *i.e.* $N_G[v] = N_G(v) \cup \{v\}$.
- $N_G(S)$: open neighborhood of a set $S \subseteq V$ *i.e.* $N_G(S) = \bigcup_{u \in S} N_G(u)$.
- $N_G[S]$: close neighborhood of a set $S \subseteq V$ *i.e.* $N_G[S] = N_G(S) \cup S$.
- $\mathrm{dist}_G(u, v)$: distance between vertices u and v *i.e.* minimum length of a path with endpoints $u, v \in V$.
- $N^i_G(v)$: set of vertices which are at distance at most i from vertex v (called i^{th} neighborhood of v) *i.e.* $N^i_G(v) = \{u \in V : \mathrm{dist}_G(v, u) \leq i\}$. Thus $N^1_G(v) = N_G(v)$.
- $G[S]$: the subgraph induced by a set $S \subseteq V$.
- $\deg_G(v)$: degree of vertex v *i.e.* $\deg_G(v) = |N_G(v)|$.
- $\Delta(G)$: maximum degree of G.
- $\mathrm{tw}(G)$: treewidth of G (see Definition 2).
- $\mathrm{ltw}_r(G)$: local treewidth of G with respect to $r \in \mathbb{N}$ (see Definition 4).
- $\mathrm{pw}(G)$: pathwidth of G (see Definition 6).
- $\mathrm{cw}(G)$: cutwidth of G.
- $\mathrm{bw}(G)$: bandwidth of G.
- $V(H)$: the set of vertices of a graph H.
- $E(H)$: the set of edges of a graph H.

We may skip the subscript or the argument if the graph G is clear from the context.

Graph classes. We briefly review the graph classes encountered throughout this work.

A *tree* is a connected graph without cycles. Let T be a tree and let r be a vertex of T designated as *root*. We say that T is a *rooted tree*. We define the *level* i of T to be the set of vertices that are at distance exactly i from r. A *leaf* is a vertex of degree one. An *ancestor* (resp. *descendant*) of a vertex v in T is any vertex on the path from r to v (resp. from v to a leaf which does not contain any ancestors of v). The *height* of T is the length of a longest path from r to a leaf. A *child* of a vertex v in T is an adjacent descendant of v. The tree T is said to be *complete* if every non-leaf vertex has exactly the same number of children. The tree T is said to be *full* if it is complete and all leaves are at the same distance from the root.

A *t-ary tree* is a rooted tree in which every vertex other than the leaves has t children.

A graph is *bipartite* if the vertices can be partitioned into two sets such that any two vertices in the same set are not adjacent.

A graph is *regular* if all the vertices have the same degree.

A *Δ-regular graph* is a regular graph where vertices have degree Δ.

A graph is *complete* (or is a *clique*) when every vertex is adjacent to every other vertex.

A graph is *planar* if it can be drawn in the plane without any edges crossing.

A *caterpillar* is a tree such that the vertices with degree at least two induce a path. In other words, a caterpillar consists of a path P such that all edges have at least one endpoint in P.

A *k-caterpillar*, for some integer $k \geq 1$, is a caterpillar in which every pending edge, *i.e.* every edge uv with exactly one endpoint, say u, in P, may be replaced by a path of length at most k. This path is then called a *leg* of the k-caterpillar at vertex u.

A *star* is a tree consisting of one vertex, called the *center* of the star, adjacent to all the other vertices.

A *k-star*, for some integer $k \geq 1$, is a tree obtained from a star in which every edge may be replaced by a path of length at most k. Notice that a k-star is a special case of a k-caterpillar.

A *cograph* is a graph that does not contain an induced P_4, that is, a path on four vertices.

An *interval graph* is a graph $G = (V, E)$ for which there exists a set of real intervals $\{I_v : v \in V\}$ such that $I_v \cap I_u \neq \emptyset$ if and only if $uv \in E$.

(Local) Treewidth. Informally speaking, the treewidth of a graph is a number that reflects how "close" the graph is to being a tree. This notion was first introduced by Halin [62] and further developed by Robertson and Seymour [98]. As a matter of fact, treewidth can be defined in several equivalent ways. Here we use the notion of tree decomposition of a graph to define it.

Definition 1: Tree decomposition

A tree decomposition of a graph $G = (V, E)$ is a pair $\mathcal{T} = (T, \mathcal{H})$ where T is a tree with vertex set X and $\mathcal{H} = \{H_x : x \in X\}$ is a family of subsets of V, such that the following conditions are met

1. $\bigcup_{x \in X} H_x = V$.
2. For each $uv \in E$ there is an $x \in X$ with $u, v \in H_x$.
3. For each $v \in V$, the set of nodes $\{x \in X : v \in H_x\}$ induces a subtree of T.

In the above definition, the third condition is equivalent to assuming that if $v \in H_{x'}$ and $v \in H_{x''}$ then $v \in H_x$ for all nodes x of the unique path from x' to x'' in T. The width of a tree decomposition $\mathcal{T}$ is $\max_{x \in X} |H_x| - 1$. We are now ready to give the definition of treewidth.

Definition 2: Treewidth

The treewidth $\mathrm{tw}(G)$ of a graph G is the minimum width over all possible tree decompositions of G.

The "-1" in the definition is included for the convenience that trees have treewidth 1 (rather than 2). It is worth noting that finding the treewidth of a graph is NP-hard [9] but can be found in linear time for graphs of bounded treewidth [22]. Concerning the appearance of substructures, one can see that the subtree T_x of T rooted at node x represents the subgraph G_x induced by precisely those vertices of G which occur in at least one H_y where y runs over the nodes of T_x.

In an algorithmic perspective, we rather use a more refined decomposition called nice tree decomposition.

Definition 3: Nice tree decomposition

A nice tree decomposition $\mathcal{T} = (T, \mathcal{H})$ of a graph G is a tree decomposition satisfying the following conditions

1. Each node of T has at most two children.
2. For each node x with two children y, z, we have $H_y = H_z = H_x$ (x is called *join node*).
3. If a node x has just one child y, then $H_x \subset H_y$ (x is called *forget node*) or $H_y \subset H_x$ (x is called *insert node*) and $||H_x| - |H_y|| = 1$.

It is not hard to show that any tree decomposition $\mathcal{T} = (T, \mathcal{H})$ of a graph can be transformed in linear time into a nice tree decomposition $\mathcal{T}' = (T', \mathcal{H}')$ of same width, with $|T'| \leq c \cdot |T|$ for some constant $c > 0$ and with $H_x \neq \emptyset$ for all $H_x \in \mathcal{H}$.

Eppstein [53] generalized the notion of treewidth by introducing the notion of bounded local treewidth. Informally speaking, a graph has bounded local treewidth if, for any vertex v, the treewidth of the induced subgraph by the r^{th} neighborhood of v is bounded by a function that solely depends on $r > 0$.

Definition 4: Local treewidth

Given an integer $r > 0$, the local treewidth of a graph $G = (V, E)$ is the number $\mathrm{ltw}_r(G)$ defined as follows

$$\mathrm{ltw}_r(G) = \max_{v \in V} \{\mathrm{tw}(G[N^r(v)])\}$$

We then say that a graph G has bounded local treewidth if there exists a function $f : \mathbb{N} \to \mathbb{N}$ such that $\mathrm{ltw}_r(G) \leq f(r)$ for all integer $r > 0$. Here are some class of graphs of bounded local treewidth.

Every graph G of bounded treewidth has bounded local treewidth since $\mathrm{ltw}_r(G) \leq \mathrm{tw}(G)$ for all $r > 0$.

Every graph of maximum degree Δ has local treewidth bounded by $\Delta(\Delta - 1)^{r-1}$ for all $r > 0$.

Finally, a planar graph has local treewidth bounded by $3r - 1$ for all $r > 0$ [19].

Pathwidth. As treewidth measure the "tree-likeness" of a graph, the pathwidth reflects how "close" the graph is to being a path. Because of this analogy, we directly give the relevant definitions.

Definition 5: Path decomposition

A path decomposition of a graph $G = (V, E)$ is a pair $\mathcal{P} = (P, \mathcal{H})$ where P is a path with vertex set X and $\mathcal{H} = \{H_x : x \in X\}$ is a family of subsets of V, such that the following conditions are met

1. $\bigcup_{x \in X} H_x = V$.
2. For each $uv \in E$ there is an $x \in X$ with $u, v \in H_x$.
3. For each $v \in V$, the set of nodes $\{x \in X : v \in H_x\}$ induces a path.

Definition 6: Pathwidth

The pathwidth $\mathrm{pw}(G)$ of a graph G is the minimum width over all possible path decompositions of G.

We also have the notion of nice path decomposition but we skip its definition here since we do not make use of it.

2.2 Decision problems & complexity

Before we give the definition of a decision problem, we first recall some basics from the theory of formal languages. An *alphabet*, denoted by Σ, is a set of symbols. A *string* (or *word*) is a finite sequence of symbols from a given alphabet. The length of a string x, denoted $|x|$, is the number of symbols in the string. The set of all strings over an alphabet Σ is denoted by Σ^*. A *language* L is a subset of Σ^*.

Definition 7: Decision problem

A *decision problem* is a language $L \subseteq \Sigma^*$ over a binary alphabet $\Sigma = \{0, 1\}$.

In fact, when we introduce a new decision problem, we will make use of the following more informal and standard way to define it.

Problem Name
Input: Some inputs
Question: A yes-no question that solely depends upon the inputs.

We do not discuss here how to encode such definition into a regular decision problem *i.e.* into a language $L \subseteq \Sigma^*$, and we assume, throughout this thesis, that we use only reasonable encoding to do so (see Garey and Johnson [60, Chapter 2.1]). More generally, the specification of any kind of problem (decision problem, parameterized problem (see Definition 11), and optimization problem (see Definition 20)) as well as integers, graphs, formulas, etc. is assumed to be encoded in binary in the usual way. Therefore, we will address these objects directly instead of working with their formal string representations.

An *instance* of a decision problem is a concrete utterance of the problem represented as a string $x \in \Sigma^*$. The size of an instance is then the size of the corresponding string. Given an instance $x \in \Sigma^*$ of a decision problem $L \subseteq \Sigma^*$, we say that x is a *yes-instance* if $x \in L$ and a *no-instance* otherwise.

P & NP classes. The subsequent definitions make use of the concept of *Turing machine*, see for instance Arora and Barak [11] for more details.

Definition 8: NP class

The class NP contains every decision problem $L \subseteq \Sigma^*$ for which the question "Does x belongs to L?" where $x \in \Sigma^*$ can be decided by a non-deterministic Turing machine that runs in polynomial time *i.e.* , the number of steps performed by the machine is upper bounded by a polynomial expression in $|x|$.

Let $L_1, L_2 \subseteq \Sigma^*$ be two decision problems. We say that L_1 polynomial-time reduces to L_2 if there exists an algorithm that takes as input an instance $x_1 \in \Sigma^*$ and outputs in polynomial time a new instance $x_2 \in \Sigma^*$ such that $x_1 \in L_1$ if and only if $x_2 \in L_2$.

A decision problem L is NP-hard if every problem of NP polynomial-time reduces to L. If a decision problem is NP-hard and is in NP then it is NP-complete.

Definition 9: P class

The class P contains every decision problem $L \subseteq \Sigma^*$ for which the question "Does x belongs to L?" where $x \in \Sigma^*$ can be decided by an algorithm that runs in polynomial time *i.e.* , the number of steps performed by the algorithm is upper bounded by a polynomial expression in $|x|$.

Asymptotic notation. In order to express the running time of an algorithm, we use the following standard notation.

Definition 10: Big O notation

Let g be a real function. We denote by $O(g(n))$ the set of all real functions f for which there exist a constant $c > 0$ and a value n_0 such that $f(n) \leq c \cdot g(n)$ for all $n > n_0$.

We might use the above notation in a more involved way. For example, we denote by $n^{O(1)}$ some function f of the form $f(n) = n^d$ where $d \in O(1)$.

2.3 Parameterized complexity

The parameterized complexity is a framework which provides a new way to express the computational complexity of decision problems. For example, consider the well known NP-complete Vertex Cover problem: given an graph $G = (V, E)$ and an integer k, determine whether there is a subset $S \subseteq V$, $|S| \leq k$, such that every edge is covered by S *i.e.*, for all $uv \in E$ we have $u \in S$ or $v \in S$. Since the problem is NP-hard it is unlikely that there is an algorithm that solves any instance of the problem in polynomial time. However, one can solve an instance using the following exponential-time algorithm: for each subset $S \subseteq V$ of vertices, pick the one that is a vertex cover with size at most k. This trivial procedure has running time $O(n^k) = O(2^{k \log n})$ where $n = |V|$. Of course this algorithm is not satisfying *i.e.* we would like to have an algorithm that does something

smarter than just trying all possibilities. One way to achieve this goal is to design a so-called *parameterized algorithm*. The general idea of this approach is to shift the exponential blowup from the input size n to a parameter of the input, *i.e.* we would like to get an algorithm with running time $f(k) \cdot n^{O(1)}$ where f is a typically exponential function that solely depends on the parameter k (see Figure 2.1). As an illustration, let us consider the Vertex Cover problem with the solution size k as the parameter. Observe that since each edge uv has to be covered, either u or v (or both) must be in the solution. Using this remark, we can solve the problem by the following recursive procedure. Let $uv \in E$ be any edge. The graph G has a vertex cover of size k if either $G \setminus v$ or $G \setminus u$ has a vertex cover of size $k-1$. It is not hard to see that there are at most k recursive calls implying a running time of $O(2^k \cdot n)$. In this case, for "small" enough value of k, we can decide an instance efficiently, no matter how large the input graph is. Indeed the running time grows linearly with the graph size. We say that Vertex Cover parameterized by k is *fixed-parameter tractable*. As a matter of fact, the best known algorithm for solving Vertex Cover takes time $1.2738^k \cdot n^{O(1)}$ [34].

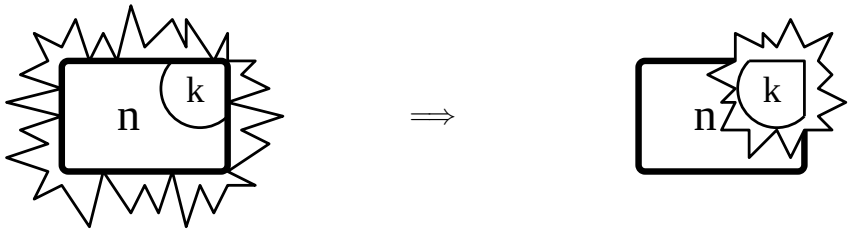


Figure 2.1: Shifting the combinatorial explosion into the parameter.

Let us define more formally the notion of parameterized problem.

Definition 11: Parameterized problem

A *parameterized problem* is a subset $Q \subseteq \Sigma^* \times \mathbb{N}$ where the first component is a decision problem and the second component is called the *parameter* of the problem.

A parameterized problem is then a decision problem $L \subseteq \Sigma^*$ where each instance is associated with some integer value $k \in \mathbb{N}$. We also say that L *is parameterized by* k. For example, a lot of decision problems are obtained from an optimization version, thus a natural and well studied parameter candidate is the solution size k also called *standard parameterization*. However, we would like to emphasize that the parameter can be something totally different and less explicit *e.g.* structural parameterization for graph problems (see Figure 2.2). Finally, it is worth noting that there exists a general definition of a parameterized problem that allows for more complicated parameters. For example, in the following problem the parameter could be the graph H.

Graph Minor Testing
Input: Two graphs H and G
Question: Does G contains H as a minor?

However, the Definition 11 is sufficient for our purpose since all parameters considered in this work are integers.

Definition 12: FPT class

The class FPT contains every parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ for which the question "Does (x, k) belongs to Q?" can be decided by an algorithm that runs in $f(k) \cdot |x|^{O(1)}$ time (or *fpt-time*) where $(x, k) \in \Sigma^* \times \mathbb{N}$ and f is a function depending solely on k.

A problem in FPT is called *fixed-parameter tractable*. We also define the class XP as follows.

Definition 13: XP

The class XP contains every parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ for which the question "Does (x, k) belongs to Q?" can be decided by an algorithm that runs in $|x|^{g(k)}$ time where $(x, k) \in \Sigma^* \times \mathbb{N}$ and g is a function depending solely on k.

2.3.1 Fixed-parameter algorithm techniques

In this section, we will review some basic tools used in this thesis for proving fixed-parameter tractability.

Kernelization. One of the main tool to devise parameterized algorithms is the *kernelization* technique. This can be regarded as a *pre-processing* over the instance of a parameterized problem. It consists of getting rid of the "easy-to-solve" part of the instance so that it only remains the *kernel* of the problem *i.e.* the "hard" part to solve. It turns out that a parameterized problem is in FPT if and only if there exists such pre-processing. Let us now define more formally this notion.

Definition 14: Kernelization

Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. A *kernelization* is an algorithm that takes as input a pair $(x, k) \in \Sigma^* \times \mathbb{N}$ and outputs in $(|x| + k)^{O(1)}$ time, a new pair $(x', k') \in \Sigma^* \times \mathbb{N}$ with $k' \leq k$ such that

1. $(x, k) \in Q \Leftrightarrow (x', k') \in Q$
2. $|x'| \leq g(k)$ and $k' \leq g(k)$ for some computable function g

The instance (x', k') is called a *kernel* of size $g(k)$. If g is a polynomial then we have a *polynomial* kernel.

The following well-known theorem shows the equivalence between a problem being fixed-parameter tractable and admitting a kernel.

Theorem 1 *A parameterized problem Q is fixed parameter tractable if and only if Q admits a kernelization.*

Proof. Let $(x, k) \in \Sigma^* \times \mathbb{N}$. Assume that Q admits a kernelization. We first apply the kernelization on (x, k) to get, in $(|x| + k)^{O(1)}$ time, a new equivalent instance $(x', k') \in \Sigma^* \times \mathbb{N}$ such that $k' \leq g(k)$ and $|x'| \leq g(k)$ for some computable function g. Next, it suffices to apply any algorithm on the kernel to solve the former instance. The overall running time is then $h(g(k)) + (|x| + k)^{O(1)}$ for some computable function h.

Conversely, suppose that Q is fixed parameter tractable and can be solved in time $f(k) \cdot |x|^c$ for some constant $c > 0$. We have to distinguish between the following

two cases

- if $|x| \leq f(k)$ then the instance itself is a kernel.
- if $|x| > f(k)$ then simply run the parameterized algorithm to solve (x, k). This is done in polynomial time since $f(k) \cdot |x|^c \leq |x|^{c+1}$. Depending of the answer, we return a trivial no-instance or yes-instance.

This completes the proof. ■

In practice, we use so-called "reduction rules" to iteratively reduce the size of an instance and finally obtain the desired kernel. A reduction rule is simply an algorithm that computes in polynomial time a new equivalent instance with reduced size. As an illustrative example, consider the following reduction rule for Vertex Cover.

Reduction rule 2 *Let (G, k) be an instance of* Vertex Cover. *If there exists a vertex v with $\deg(v) > k$ then remove v from G and decrease k by one.*

After applying iteratively the above reduction rule, one gets a new equivalent instance (G', k') where G' has size n'. Indeed, observe that if a vertex v has degree larger than k then it must be in the solution, otherwise we would have taken all its neighbors to cover all edges. Thus, we can safely remove v from G and decrease k by one to reflect the fact that v is part of any vertex cover. Now, suppose that there is a vertex cover S of size k' for G'. Thus, we must have $n' \leq k^2$ since S covers every edge and the maximum degree of G' in k. The kernelization is then defined as follows. If $n' > k^2$ then return any trivial no-instance, otherwise we have $n' \leq k^2$ giving us a polynomial kernel.

Monadic Second Order Logic (MSOL). The *monadic second order logic* is another powerful tool to prove fixed parameter tractability with respect to parameter treewidth (see Definition 2). This logic is an extension of the *first order logic* which is, in turns, an extension of the *propositional logic*. The key point here is that whenever a graph problem can be expressed as a MSO-formula, we can derive the result that it is in FPT with respect to parameter treewidth. The language of MSOL for graphs includes the logical connectives $\vee$, $\wedge$, $\neg$, $\leftrightarrow$, $\rightarrow$, variables for vertices, edges, sets of vertices, and sets of edges, the quantifiers $\forall$, $\exists$ that can be applied to these variables, and the following four binary relations:

1. $u \in U$, where u is a vertex variable and U is a vertex set variable.
2. $d \in D$, where d is an edge variable and D is an edge set variable.
3. $\mathrm{adj}(u, v)$, where u, v are vertex variables, and the interpretation is that u and v are adjacent.
4. Equality, $=$, of variables representing vertices, edges, sets of vertices, and sets of edges.

Given a formula ϕ, we denote by $G \models \phi$ the fact that the graph G satisfies the property ϕ — we also say that G is a *model* of ϕ. For example the following statement,

$$G \models \forall x(x \in V \rightarrow (\exists y(y \in V \wedge y \in P \wedge \mathrm{adj}(x, y))) \vee x \in P)$$

holds if and only if the set P is a dominating set in the graph G.

For our purpose, the central result of this theory is as follows.

Theorem 3 (Courcelle [44]) *Let G be a graph and ϕ a MSO formula of length ℓ. The problem of deciding whether $G \models \phi$ is fixed parameter tractable with respect to the combined parameter $\mathrm{tw}(G)$ and ℓ.*

2.3.2 Kernelization lower bounds

As previously hinted, any fixed parameter problem admits a kernel of size depending of the running time of the parameterized algorithm. This arises the natural question whether there always exists a "small" kernel *i.e.* a kernel of polynomial size. We observed that it is the case for the Vertex Cover problem. Unfortunately, it is not always possible to obtain such kernel for all problems in FPT. Bodlaender et al. [20] introduced a technique for proving that a parameterized problem does not admit a polynomial kernel unless a well-established complexity theory assumption unexpectedly collapses. This technique is based on the previous results of Bodlaender et al. [20] and Fortnow and Santhanam [59]. We first recall here the crucial definitions.

Definition 15: Polynomial equivalence relation [21]

An equivalence relation $\mathcal{R}$ on Σ^* is called a *polynomial equivalence relation* if the following conditions are met

1. There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether $\mathcal{R}(x, y)$ in $(|x| + |y|)^{O(1)}$ time.
2. For any finite set $S \subseteq \Sigma^*$ the equivalence relation $\mathcal{R}$ partitions the elements of S into at most $(\max_{x \in S} |x|)^{O(1)}$ classes.

Definition 16: Cross-composition [21]

Let $L \subseteq \Sigma^*$ be a decision problem and $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. We say that L *cross-composes* into Q if there is a polynomial equivalence relation $\mathcal{R}$ and an algorithm which, given t strings $x_1, x_2, \ldots x_t$ belonging to the same equivalence class of $\mathcal{R}$, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbb{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that

1. $(x^*, k^*) \in Q \Leftrightarrow x_i \in L$ for some $1 \leq i \leq t$.
2. k^* is bounded polynomially in $\max_{i=1}^{t} |x_i| + \log t$.

The central result for proving kernel size lower bound is the following.

Theorem 4 ([21], Theorem 9) *Let $L \subseteq \Sigma^*$ be a decision problem, and let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. If L is NP-hard and cross-composes into Q then Q that has no polynomial kernel unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$.

2.3.3 Fixed-parameter intractability

Parameterized complexity theory also provides methods for proving the inherent intractability of a parameterized problem. To this end, we need to introduce the notion of *parameterized reduction (or fpt-reduction)* defined as follows.

Definition 17: Parameterized reduction

Let $Q_1, Q_2 \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. We say that Q_1 fpt-reduces to Q_2 if there exists an algorithm that takes as input an instance $(x_1, k_1) \in \Sigma^* \times \mathbb{N}$ and outputs in $f(k_1) \cdot |x_1|^{O(1)}$ time a new instance $(x_2, k_2) \in \Sigma^* \times \mathbb{N}$ such that

1. $(x_1, k_1) \in Q_1 \Leftrightarrow (x_2, k_2) \in Q_2$
2. $k_2 \leq g(k_1)$

for some computable functions f and g.

Before introducing the different classes of complexity, we need to define the concept of *boolean circuit*.

Definition 18: Boolean circuit

A boolean circuit $\mathcal{C} = (V, A)$ is a directed acyclic graph whose vertices V are called *gates*. The gates of in-degree 0 are called *inputs*. There is exactly one gate of out-degree 0 called *output*. Every gate that is neither an input nor an output is labeled by an element of $\{OR, AND, NOT\}$. A gate with label NOT has in-degree exactly one.

We need to define some additional terminology. Let $\mathcal{C} = (V, A)$ be a boolean circuit. A gate with in-degree bounded by some fixed constant is said *small* and *large* otherwise. The *weft* of $\mathcal{C}$ is the maximum number of large gates on a path from an input to the output. The *depth* is the maximum number of all gates on a path from an input to the output. A *truth assignment* for $\mathcal{C}$ is a function $\tau : V \to \{true, false\}$ that associates the value $true$ or $false$ to each input gates. The (Hamming) *weight* of a truth assignment is the number of input gates set to $true$. Given an assignment τ for $\mathcal{C}$, the value $\nu(g)$ of a gate g is recursively defined as follows

$$\nu(g) = \begin{cases} \tau(g) & \text{if } g \text{ is an input} \\ \bigvee_{h \in N^-(g)} \nu(h) & \text{if } g \text{ is labeled by OR} \\ \bigwedge_{h \in N^-(g)} \nu(h) & \text{if } g \text{ is labeled by AND} \\ \neg\nu(h) & \text{if } g \text{ is labeled by NOT with } N^-(g) = \{h\} \end{cases}$$

A truth assignment satisfies $\mathcal{C}$ if the value of the output gate is $true$. We can now introduce the following central problem that is used to define the hierarchies of parameterized complexity classes.

Weft-t Circuit Satisfiability
Input: A boolean circuit $\mathcal{C}$ with constant depth and weft at most t and an integer k.
Question: Is there a truth assignment of weight k that satisfies $\mathcal{C}$?

Definition 19: W[t] class

A parameterized problem $Q \subseteq \Sigma^* \times \mathbb{N}$ belongs to W[t] , for fixed $t > 0$, if Q fpt-reduces to Weft-t Circuit Satisfiability parameterized by k.

The inclusion relationship of the above classes are as follows.

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \ldots \subseteq \mathsf{XP}$$

Informally speaking, a problem in W[t] is considered "harder" than those lying in W[t-1] where $t > 1$. We say that a parameterized problem is W[t]-hard if every problem of W[t] fpt-reduces to it. If the problem moreover belongs to W[t] then it is W[t]-complete. There is a good reason to believe that W[t]-hard problems are unlikely to be in FPT. It is worth pointing out that, in practice, we only consider W[1] and W[2] as the basic classes of parameterized intractability.

Cesati [30] introduced a more "classical" way to prove that a given parameterized problem belongs to the class W[1] (or W[2]) by the use of Turing machine. For that purpose, we need to introduce the following two problems.

SHORT NONDETERMINISTIC TURING MACHINE
Input: A nondeterministic Turing machine M, a string x on the input alphabet of M, and an integer k.
Question: Is there a computation of M on input x that reaches a final accepting state in at most k steps?

SHORT MULTI-TAPE NONDETERMINISTIC TURING MACHINE
Input: A multi-tape nondeterministic Turing machine M, a string x on the input alphabet of M, and an integer k.
Question: Is there a computation of M on input x that reaches a final accepting state in at most k steps?

We have now the following result due to Cesati [30].

Theorem 5 *A parameterized problem is in* W[1] *(resp.* W[2]*) if it can be fpt-reduced to the* SHORT NONDETERMINISTIC TURING MACHINE *(resp.* SHORT MULTI-TAPE NONDETERMINISTIC TURING MACHINE*) problem parameterized by k.*

Observe that the previous theorem plays the same role as the non-deterministic Turing machine does for the class NP.

2.3.4 Parameters hierarchies

We conclude this section on parameterized complexity with the following useful property.

Theorem 6 *Let $Q_1, Q_2 \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. If the parameters k_1 of Q_1 and k_2 of Q_2 always satisfy the inequality $k_1 \leq c \cdot k_2$ for some constant $c > 0$ then the following assertions are true.*

1. *If Q_2 is* W[t]*-hard (resp.* NP*-hard for constant values of k_2) for some integer $t > 0$ then Q_1 is* W[t]*-hard (resp.* NP*-hard for constant values of k_1).*
2. *If Q_1 is in* FPT *(resp.* XP*) then Q_2 is in* FPT *(resp.* XP*).*

As an application of the above theorem, consider the Figure 2.2. If a parameterized problem is in FPT (resp. W[t]-hard, NP-hard for constant value of k) with respect to some structural parameter k then, using Theorem 6, it is in FPT (resp. W[t]-hard, NP-hard for constant value of k) with respect to each parameter which is an ancestor (resp. descendant) of k. A parameter k_1 is an ancestor (resp. descendant) of a parameter k_2 if there exists a directed path from k_1 to k_2 (resp. from k_2 to k_1).

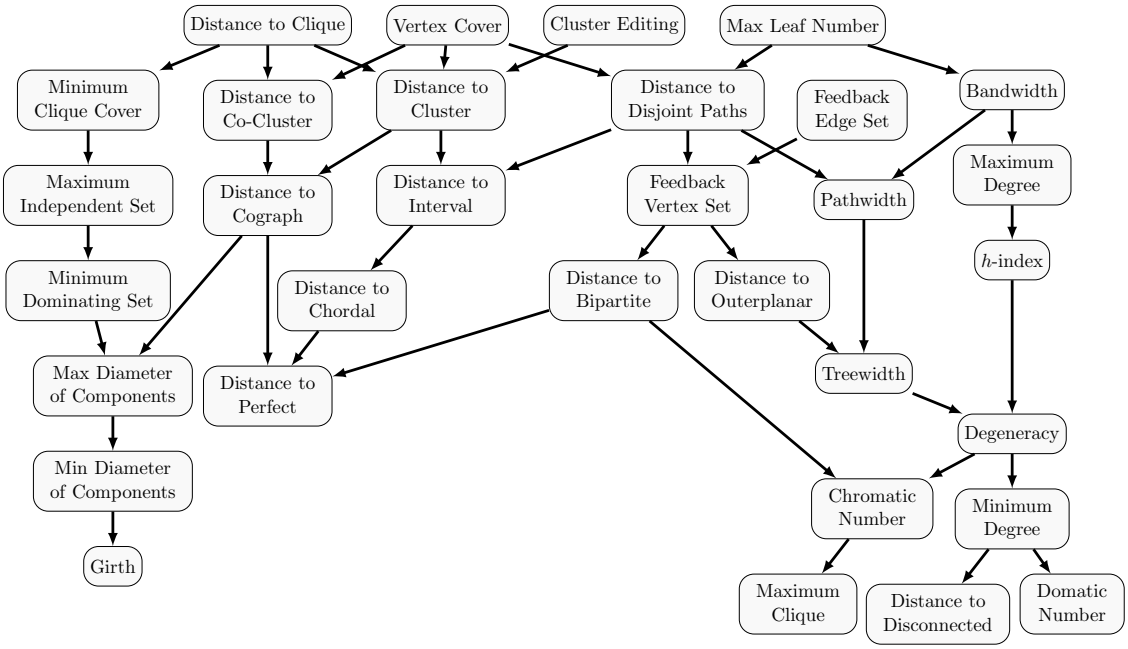


Figure 2.2: Overview of the relations between some structural graph parameters, see the paper of Komusiewicz and Niedermeier [77] for more details. An arc from a parameter k_2 to a parameter k_1 means that there exists a constant $c > 0$ such that $k_1 \leq c \cdot k_2$. Given a graph G, the "Distance to $\mathcal{C}$" parameter where $\mathcal{C}$ is a class of graphs, corresponds to the minimum number of vertices to remove from G in order to obtain a new graph that belongs to $\mathcal{C}$.

2.4 Approximation

Another approach for coping with NP-hard problems is to use the approximation framework. Indeed, computing an optimal solution might not be mandatory and could be a really hard task even with time efficient algorithm. That is where the algorithms with performance guarantee come in. In such algorithms, the basic idea is to "release" the constraint of optimality while the running time is required to be reasonable. Moreover, the computed solution must be not "too far" from the optimum. In what follows we define more formally these notions.

2.4.1 Optimization problem

So far, we have dealt with decision problems. In the context of approximation algorithms, we rather consider *optimization* problems. While in a decision problem the solution is either "yes" or "no", an optimization problem asks to find a solution that maximizes (or minimizes) an objective function.

Definition 20: Optimization problem & NPO class

An optimization problem O is a 4-tuple $(D, \text{sol}, \text{cost}, \text{goal})$ where

1. $D \subseteq \Sigma^*$ is the set of instances recognizable in polynomial time.
2. For each instance $x \in D$, $\text{sol}(x) \in \mathcal{P}(\Sigma^*)$ is the set of feasible solutions of x, where $\mathcal{P}(X)$ is the set of all subsets of a set X.

3. The size of each solution $y \in \text{sol}(x)$ is polynomially bounded in $|x|$ *i.e.* $|y| \leq |x|^{O(1)}$.
4. It can be decided in polynomial time whether $y \in \text{sol}(x)$ holds for given x and y.
5. Given an instance x and a feasible solution y, $\text{cost}(x, y)$ is a polynomial-time computable positive integer.

The class that contains all optimization problems is denoted by NPO .
A *minimization* (resp. *maximization*) problem is an optimization problem with goal = min (resp. goal = max).

As a matter of fact, the above definition stands for an NP optimization problem which is a particular optimization problem. However, this thesis only deals with optimization problems that are NP optimization problems, allowing us the use of this shortcut.

Given an instance x of some optimization problem, the goal is to find an optimal solution, that is, a feasible solution $y \in \text{sol}(x)$ such that

$$\text{cost}(x, y) = \underset{y' \in \text{sol}(x)}{\text{goal}} \{\text{cost}(x, y')\}$$

The cost of an optimum solution for an instance x is denoted by $\text{opt}(x)$.

As an illustration, the Max Clique problem is to find a complete subgraph of a given graph of maximum size (see Appendix A). Formally, this problem is defined as follows.[1]

- $D = \{G = (V, E) : G \text{ is a graph}\}$.
- Let G be any graph, $\text{sol}(G) = \{H = (V', E') : H \text{ is a complete subgraph of } G\}$
- Let G be any graph and H a complete subgraph of G of size n, $\text{cost}(G, H) = n$.
- goal = max

In this thesis, when a new optimization problem will be introduced, we will make use of the following more convenient definition.

Problem Name
Input: Some inputs
Output: Objective to achieve.

An optimization problem is polynomial time solvable if there exists an algorithm that computes, for every instance, an optimal solution within a running time polynomial in the instance size.

Definition 21: PO class

The class PO contains all problems of NPO that are polynomial-time solvable.

The link between optimization problems and decision problems is the following. If $O = (D, \text{sol}, \text{cost}, \text{goal})$ is a maximization problem then we can define its decision version as follows (for minimization problems the definition is analogous).

[1] As hinted in Section 2.2, we do not discuss how to actually encode objects into strings of Σ^*.

O_k
Input: An instance $x \in D$ and an integer k.
Question: Is $\text{opt}(x) \geq k$?

In fact the decision version of a optimization problem is in NP. In general, we are interested in optimization problems for which the decision version is NP-hard. By definition, an optimization problem is NP-hard if its decision version is NP-hard.

2.4.2 Approximation algorithms

Before defining what an approximation algorithm is, we need to determine how to measure the quality of a solution. For that purpose, we introduce the performance ratio.

Definition 22: Performance ratio & error

Given an instance x of an optimization problem $O = (D, \text{sol}, \text{cost}, \text{goal})$, the performance ratio $r(x, y)$ of a solution $y \in \text{sol}(x)$ is

$$r(x, y) = \max\left\{\frac{\text{cost}(x, y)}{\text{opt}(x)}, \frac{\text{opt}(x)}{\text{cost}(x, y)}\right\}$$

The *error* of y, denoted $\varepsilon(x, y)$, is defined by $\varepsilon(x, y) = r(x, y) - 1$.

Notice that the performance ratio is always > 1 and it gets closer to 1 as the solution gets closer to the optimum.

Definition 23: α-approximation algorithm

Let $O = (D, \text{sol}, \text{cost}, \text{goal})$ be an optimization problem and $\alpha : \mathbb{N} \to]1, +\infty[$ be a function. An α-approximation algorithm is an algorithm that takes as input any instance $x \in D$ of size $n = |x|$ and returns a solution $y \in \text{sol}(x)$ such that $r(x, y) \leq \alpha(n)$.

A problem admitting an α-approximation algorithm is said to be α-approximable. Notice that we did not impose any specific running time in the above definition. A polynomial-time α-approximation algorithm is an α-approximation algorithm with running time polynomial in the instance size. Let F be a set of functions, we denote by F-APX the class of optimization problems for which there exists such algorithm with $\alpha \in F$. We now review the different types of approximation classes encountered throughout this thesis.

Definition 24: APX, *log*-APX and *poly*-APX classes

The class APX (resp. *log*-APX, *poly*-APX) corresponds to the class F-APX where $F = O(1)$ (resp. $O(\log(n))$, $O(n^c)$ for some fixed constant $c > 0$).

Definition 25: PTAS class

An optimization problem admits a polynomial-time approximation scheme (ptas for short) if for any fixed constant $\varepsilon > 0$, there exists a polynomial-time $(1 + \varepsilon)$-approximation algorithm for that problem.
The class PTAS contains all optimization problems that admit a ptas.

Generally, the running time of a ptas is exponential in $1/\varepsilon$, like $n^{O(1/\varepsilon)}$ if the size of the instance is n. We have the following inclusions

$$\mathsf{PO} \subseteq \mathsf{PTAS} \subseteq \mathsf{APX} \subseteq \mathit{log}\text{-}\mathsf{APX} \subseteq \mathit{poly}\text{-}\mathsf{APX} \subseteq \mathsf{NPO}$$

We conclude this section by introducing the concept of parameterized approximation. We first extend the definition of an optimization problem as follows.

Definition 26: Parameterized optimization problem

A *parameterized optimization problem* is an optimization problem where each instance is associated with a nonnegative integer called *parameter*. We denote such optimization problem by $O^p = (D^p, \text{sol}, \text{cost}, \text{goal})$ where $D^p \subseteq \Sigma^* \times \mathbb{N}$.

We are now ready to give the following definition.

Definition 27: fpt-time α-approximation algorithm

Let $O^p = (D^p, \text{sol}, \text{cost}, \text{goal})$ be a parameterized optimization problem and $\alpha : \mathbb{N} \to]1, +\infty[$ a function. An fpt-time α-approximation algorithm is an algorithm that takes as input any instance $(x, k) \in D^p$ of size $n = |x|$ and returns a solution $y \in \text{sol}(x)$ such that $r(x, y) \leq \alpha(n)$ in $f(k) \cdot n^{O(1)}$ time for some computable function f that solely depends on k.

We establish in [15] the following simple but useful lemma.

Lemma 7 *Let O^p be a parameterized optimization problem. If there is an fpt-time α_1-approximation algorithm for O^p and **some** strictly increasing function α_1 depending solely on the parameter then there is also an fpt-time α_2-approximation algorithm for O^p and **any** strictly increasing function α_2 depending solely on the instance size.*

Proof. Let α_1^{-1} and α_2^{-1} be the inverse functions of α_1 and α_2, respectively. Let $(x, k) \in D^p$ be an instance of a parameterized maximization problem $O^p = (D^p, \text{sol}, \text{cost}, \text{goal})$ (the proof is analogous for minimization problems). We distinguish the following two cases.

Case 1: $k \leq \alpha_1^{-1}(\alpha_2(|x|))$. In this case, we apply the α_1-approximation algorithm and directly get a solution $y \in \text{sol}(x)$ such that $r(x, y) \leq \alpha_1(k) \leq \alpha_1(\alpha_1^{-1}(\alpha_2(|x|))) = \alpha_2(|x|)$ in time $f(k) \cdot |x|^{O(1)}$ for some computable function f.

Case 2: $k > \alpha_1^{-1}(\alpha_2(|x|))$. We then have $|x| < \alpha_2^{-1}(\alpha_1(k))$ and thus we can solve x by exhaustively checking every solution y in $\text{sol}(x)$ and return the one with the largest $\text{cost}(x, y)$ value. Since $|y| \leq |x|^{O(1)}$ by the definition of an optimization problem, we know that there are at most $|\Sigma|^{|x|^{O(1)}} \leq |\Sigma|^{\alpha_2^{-1}(\alpha_1(k))^{O(1)}}$ different solutions in the set $\text{sol}(x)$. It follows that the running time in this case is $O(|\Sigma|^{\alpha_2^{-1}(\alpha_1(k))^{O(1)}}) = f(k)$ for some computable function f. This completes the proof. ■

As an illustration of this lemma, if a problem admits a polynomial-time k-approximation then we can approximate this problem within any arbitrarily small ratio depending on the instance size in fpt-time *e.g.* $\log(\log(\ldots \log(|x|))$.

2.4.3 Hardness of approximation

The main tool used in this thesis for showing directly the inapproximability of an optimization problem is the *gap-introducing* reduction.

Definition 28: Gap-introducing reduction

Let $L \subseteq \Sigma^*$ be a decision problem and $O = (D, \text{sol}, \text{cost}, \text{goal})$ be a maximization (resp. minimization) problem. We say that there is a gap-introducing reduction from L to O if there exist two functions $\alpha : \mathbb{N} \to]1, +\infty[$ and $f : D \to \mathbb{Q}$ together with an algorithm that takes as input an instance $x \in \Sigma^*$ and outputs in polynomial time a new instance $x' \in D$ such that the following conditions are met

1. if x is a yes-instance then $\text{opt}(x') \geq f(x')$ (resp. $\text{opt}(x') \leq f(x')$).
2. if x is a no-instance then $\text{opt}(x') < \frac{f(x')}{\alpha(|x'|)}$ (resp. $\text{opt}(x') > \alpha(|x'|) \cdot f(x')$).

The function α is called the *gap* or *hardness factor*. We have the following result.

Theorem 8 *If there is a gap-introducing reduction from a* NP*-hard decision problem to an optimization problem with gap α, then there is no polynomial-time α-approximation algorithm for that optimization problem unless* P = NP*.*

For proving that a given parameterized optimization problem is unlikely to have a fpt-time approximation algorithm, we need a slightly different version of the above reduction.

Definition 29: Parameterized gap-introducing reduction

Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem and $O^p = (D^p, \text{sol}, \text{cost}, \text{goal})$ be a parameterized maximization (resp. minimization) problem. We say that there is a parameterized gap-introducing reduction from Q to O^p if there exist two functions $\alpha : \mathbb{N} \to]1, +\infty[$ and $f : \Sigma^* \to \mathbb{Q}$ together with an algorithm that takes as input an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ and outputs, in $g(k) \cdot |x|^{O(1)}$ time for some computable function g, a new instance $(x', k') \in D^p$ such that the following conditions are met

1. $k' \leq h(k)$ for some computable function h that depends solely on k.
2. if $(x, k) \in Q$ then $\text{opt}(x') \geq f(x')$ (resp. $\text{opt}(x') \leq f(x')$).
3. if $(x, k) \notin Q$ then $\text{opt}(x') < \frac{f(x')}{\alpha(|x'|)}$ (resp. $\text{opt}(x') > \alpha(|x'|) \cdot f(x')$).

We then have a similar result as the one above.

Theorem 9 *If there is a parameterized gap-introducing reduction from a* W[t]*-hard parameterized problem, $t > 0$, to a parameterized optimization problem with gap α, then there is no fpt-time α-approximation algorithm for that optimization problem unless* FPT = W[t]*.*

Proof. Let $Q \subseteq \Sigma^* \times \mathbb{N}$ be a W[t]-hard parameterized problem and $O^p = (D^p, \text{sol}, \text{cost}, \text{goal})$ a parameterized maximization problem (the proof is analogous for the minimization case). Suppose that there is a parameterized gap-introducing reduction from Q to O^p with gap α and a fpt-time α-approximation algorithm A for O^p. We show that we can solve every instance (x, k) of Q in fpt-time with respect to k implying FPT = W[t].

Since there is a parameterized gap-reduction from Q to O^p, there must exist an algorithm that maps the instance (x, k) into an instance $(x', k') \in D^p$ in fpt-time with respect to k. Now we run the approximation algorithm A onto (x', k') to get a solution $y \in \text{sol}(x')$ such that $\frac{\text{opt}(x')}{\text{cost}(x', y)} \leq \alpha(|x'|)$ in fpt-time with respect to k' and thus in fpt-time with respect to k (since $k' \leq h(k)$ for some function h).

By the definition of the reduction, if (x, k) is a yes-instance then $\text{opt}(x') \geq f(x')$ and then $\frac{f(x')}{\alpha(|x'|)} \leq \text{cost}(x', y)$. Conversely, if (x, k) is a no-instance then $\text{opt}(x') < \frac{f(x')}{\alpha(|x'|)}$ and thus $\text{cost}(x', y) < \frac{f(x')}{\alpha(|x'|)}$ (since $\text{cost}(x', y) \leq \text{opt}(x')$). It follows that by comparing the values $\text{cost}(x', y)$ and $\frac{f(x')}{\alpha(|x'|)}$, we are able to distinguish between yes- and no-instances of Q in fpt-time with respect to k. This completes the proof. ■

2.4.4 Approximation preserving reductions

For proving that a problem is at least as hard to approximate as another, we need to introduce the notion of *approximation preserving reduction*. This kind of reduction is more involved than the one to prove NP-hardness for decision problems. This is essentially because we have to deal with quantitative solutions and not simply a "yes" or "no" solution. Although the literature references many different approximation reductions [12], we only present the L-reduction and E-reduction in this thesis. We start with the L-reduction ("linear reduction") introduced by Papadimitriou and Yannakakis [92].

Definition 30: L-reduction

Let $O_1 = (D_1, \text{sol}_1, \text{cost}_1, \text{goal}_1)$ and $O_2 = (D_2, \text{sol}_2, \text{cost}_2, \text{goal}_2)$ be two optimization problems. We say that O_1 is L-reducible to O_2 if there are two constants $\alpha, \beta > 0$ and two polynomial time computable functions f, g such that

1. f maps each instance $x_1 \in D_1$ into an instance $x_2 \in D_2$ such that $\text{opt}_2(x_2) \leq \alpha \cdot \text{opt}_1(x_1)$.
2. g maps each solution $y_2 \in \text{sol}_2(x_2)$ into a solution $y_1 \in \text{sol}_1(x_1)$ such that $|\text{cost}_1(x_1, y_1) - \text{opt}_1(x_1)| \leq \beta \cdot |\text{cost}_2(x_2, y_2) - \text{opt}_2(x_2)|$.

An optimization problem is APX-hard if every problem of APX L-reduces to that problem. An APX-hard optimization problem is unlikely to be in PTAS, just as NP-hard decision problems are unlikely to be in P. Furthermore, if a problem O_1 is L-reducible to a problem O_2 then the following holds: If O_1 is APX-hard then O_2 is also APX-hard and if $O_2 \in$ PTAS then $O_1 \in$ PTAS. Unfortunately, this property is no longer true for a class beyond APX. For instance, let $\varepsilon \in (0, 1)$ be any fixed constant, if a problem O_1 is $n^{1-\varepsilon}$-APX-hard and L-reduces to another problem O_2 then we cannot deduce that O_2 is also $n^{1-\varepsilon}$-APX-hard. To get rid of this problem, we need to use the E-reduction ("error-preserving reduction") introduced by Khanna et al. [73].

Definition 31: E-reduction

Let $O_1 = (D_1, \text{sol}_1, \text{cost}_1, \text{goal}_1)$ and $O_2 = (D_2, \text{sol}_2, \text{cost}_2, \text{goal}_2)$ be two optimization problems. We say that O_1 is E-reducible to O_2 if there exist polynomial time computable functions f, g and a constant β such that

1. f maps each instance $x_1 \in D_1$ into an instance $x_2 \in D_2$ such that $\text{opt}_1(x_1)$ and $\text{opt}_2(x_2)$ are related by a polynomial factor, *i.e.* there exists a polynomial p such that $\text{opt}_2(x_2) \leq p(|x_1|) \cdot \text{opt}_1(x_1)$.
2. g maps each solution $y_2 \in \text{sol}_2(x_2)$ into a solution $y_1 \in \text{sol}_1(x_1)$ such that $\varepsilon(x_1, y_1) \leq \beta\varepsilon(x_2, y_2)$.

An important property of an E-reduction is that it can be applied uniformly to all levels of approximability. In other words, if a problem O_1 is E-reducible to a problem

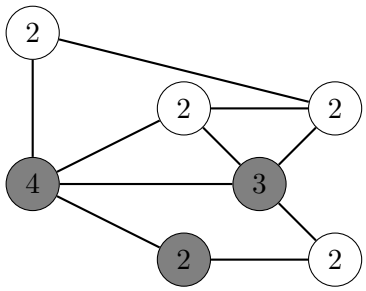
O_2 and $O_2 \in \mathcal{C}$ then $O_1 \in \mathcal{C}$, where $\mathcal{C}$ is either PTAS or F-APX. Moreover, if O_1 is F-APX-hard then O_2 is also F-APX-hard.

Chapter

3 Maximizing the spread of influence

This chapter essentially addresses the computational complexity of the problem of finding the most influential individuals in a network. Formally, given a graph and a threshold value $\text{thr}(v)$ associated to each vertex v, the task is to find and activate a vertex subset of minimum size such that all of the vertices become active at the end of the propagtion propagation process defined as follows. A vertex becomes active if at least $\text{thr}(v)$ neighbors of v are active. The propagation process proceeds in several steps and stops when no further vertex becomes active.

In the graph below, one could select the gray vertices into a target set to activate the entire graph within three steps. The numbers inside the vertices denote the thresholds.



In this chapter, we investigate the parameterized complexity and parameterized approximability of this problem and its complementary version *i.e.* given an integer k, find and activate a vertex subset of size at most k that maximizes the total number of activated vertices at the end of the propagation process.

The content of this chapter is based on the following papers.

▶ M. Chopin, A. Nichterlein, R. Niedermeier, and M. Weller, *Constant Thresholds Can Make Target Set Selection Tractable*, Proceedings of the 1st Mediterranean Conference on Algorithms (MedALG 2012), LNCS 7659, pp. 120–133, 2012.

▶ C. Bazgan, M. Chopin, A. Nichterlein and F. Sikora, *Parameterized Approximability of Maximizing the Spread of Influence in Networks*, Proceedings of the 19th Annual

International Computing and Combinatorics Conference (COCOON 2013), LNCS 7936, pp. 543–554, 2013.

3.1 Introduction

Optimization problems that involve a diffusion process in a graph are widely studied [93, 72, 35, 1, 51, 32, 17, 95]. Such problems share the common property that, according to a specified *propagation rule*, a chosen subset of vertices activates all or a fixed fraction of the vertices, where initially all but the chosen vertices are inactive. Such optimization problems model the spread of influence or information in social networks via word-of-mouth recommendations, of diseases in populations, or of faults in distributed computing [93, 72, 51]. One representative problem that appears in this context is the influence maximization problem introduced by Kempe et al. [72]. Given a directed graph, the task is to choose a vertex subset of size at most a fixed number such that the number of activated vertices at the end of the propagation process is maximized. The authors show that the problem is polynomial-time $(\frac{e}{e-1}+\varepsilon)$-approximable for any $\varepsilon > 0$ under some stochastic propagation models, but NP-hard to approximate within a ratio of $n^{1-\varepsilon}$ for any $\varepsilon > 0$ for general propagation rules. In this thesis, we rather use the following deterministic propagation model. We are given a graph, a threshold value $\mathrm{thr}(v)$ associated to each vertex v, and the following propagation rule: a vertex becomes active if at least $\mathrm{thr}(v)$ neighbors of v are active. The propagation process proceeds in several steps and stops when no further vertex becomes active. A subset S of vertices is called a target set if all the vertices of the graph get active when the vertices of S are initially activated. Given this model and an integer $k > 0$, determining the existence of a target set of size at most k is known as the Target Set Selection problem [35]. The minimization version is denoted by Min Target Set Selection and consists of finding a target set of minimum size.

Chen [35] showed hardness of approximating this last problem within a ratio $O(2^{\log^{1-\varepsilon} n})$ for any $\varepsilon > 0$, even for constant degree graphs with thresholds at most two and for general graphs when the threshold of each vertex is half its degree (later called majority thresholds). When the threshold of each vertex equals its degree (later called unanimity thresholds), the problem turns out to be polynomial-time equivalent to the vertex cover problem [35] and, thus, admits a 2-approximation, is hard to approximate within a ratio better than 1.36 [46], and the decision version is fixed-parameter tractable with respect to the parameter k. Ben-Zwi et al. [17] obtained a W[1]-hardness result with respect to the parameter "treewidth" of the input graph. However, Target Set Selection is polynomial-time solvable on graphs of bounded treewidth with the degree of the polynomial depending on the treewidth. They also proved fixed-parameter tractability for the same parameter once the threshold values are bounded by any constant. Recently, further parameterized complexity studies for the structural graph parameters "diameter", "cluster editing number", "vertex cover number", and "feedback edge set number" have been undertaken [90]. Moreover, polynomial-time algorithms for Target Set Selection restricted to special graph classes including chordal graphs and block-cactus graphs have been developed [28, 39, 95]. Finally, there are numerous combinatorial studies concerning the sizes of optimal target sets (upper and lower bounds) mostly with respect to special graph classes [2, 3, 4, 32, 39, 51, 96]. The role of the threshold values and threshold functions has been studied in the past. For instance, Dreyer and Roberts [51] showed NP-hardness for Target Set Selection when all vertices have the same threshold t, $t \geq 3$. Later, Chen [35] extended the previous result to $t = 2$. Centeno et al. [28] and Chiang et al.

[39] exploited threshold values being upper-bounded by two to develop polynomial-time algorithms for Target Set Selection on chordal graphs.

From the previous works, it can be seen that Target Set Selection is computationally hard from both a parameterized and an approximation point of view. In light of this fact, we tackle the complexity of this problem using two different approaches. First, we investigate the parameterized complexity of the problem using parameters related to the structure of the graph. Indeed, the structural parameters of some social networks might be small and/or have useful properties that help in solving the problem. For instance networks modeling romantic relationship (see [52, Chap. 2, Fig. 2.7] and Figure 1.1) are sparse and thus parameters related to the sparseness of the input graph are of particular interest. Moreover, we consider for each parameter three types of thresholds: unanimity, majority and thresholds bounded by a constant. One might think that assuming thresholds to be constant bounded is quite a restrictive hypothesis. However in several applications, such as viral marketing, it is conceivable that constant thresholds suffice to model the underlying application scenarios. For instance, independent of my total number of friends it may suffice that at least five of my friends in a social network buy a certain product to convince me about the product's usefulness. Overall, our results are pictorially summarized in Figure 3.1. Second, we combine approximation with parameterized complexity theory toward the goal of getting positive approximation results. More specifically, we turn our attention to the complementary version of Target Set Selection called Influence. This problem is defined similarly except that the input is extended with an integer $\ell > 0$ and we ask the existence of a subset of vertices of size k to activate such that at least ℓ vertices are activated at the end of the propagation process. We then consider the parameterized approximability of the maximization version where the objective is to find a subset of vertices of size k to activate that maximizes the total number of activated vertices at the end of the propagation process. This is actually the deterministic version of the problem by Kempe et al. [72]. In that problem, there are two possibilities of measuring the value of a solution: counting the vertices activated by the propagation process including or excluding the initially chosen vertices. This leads to two versions of the problem denoted by Max Closed Influence and Max Open Influence, respectively. Observe that whether or not counting the chosen vertices might change the approximation factor. In this chapter, we consider both cases and our approximability results are summarized in Table 3.1.

		Max Open Influence		Max Closed Influence	
thr	Bounds	poly-time	fpt-time	poly-time	fpt-time
Cons.	Upper Lower	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$ (Th.27)
Maj.	Upper Lower	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$	n $n^{1-\varepsilon}, \forall \varepsilon > 0$ (Th.26)
Una.	Upper Lower	2^k (Th.31) $n^{1-\varepsilon}, \forall \varepsilon > 0$ (Th.30)	$\alpha(n), \forall \alpha$ (Th.32) ?	2^k $1+\varepsilon$ (Th.36)	$\alpha(n), \forall r$?

Table 3.1: Table of our approximation results for Max Open Influence and Max Closed Influence for constant (Cons.), majority (Maj.) and unanimity (Una.) thresholds. The fpt-time is with respect to the parameter k for both problems. In this table, the value n corresponds to the size of the input graph.

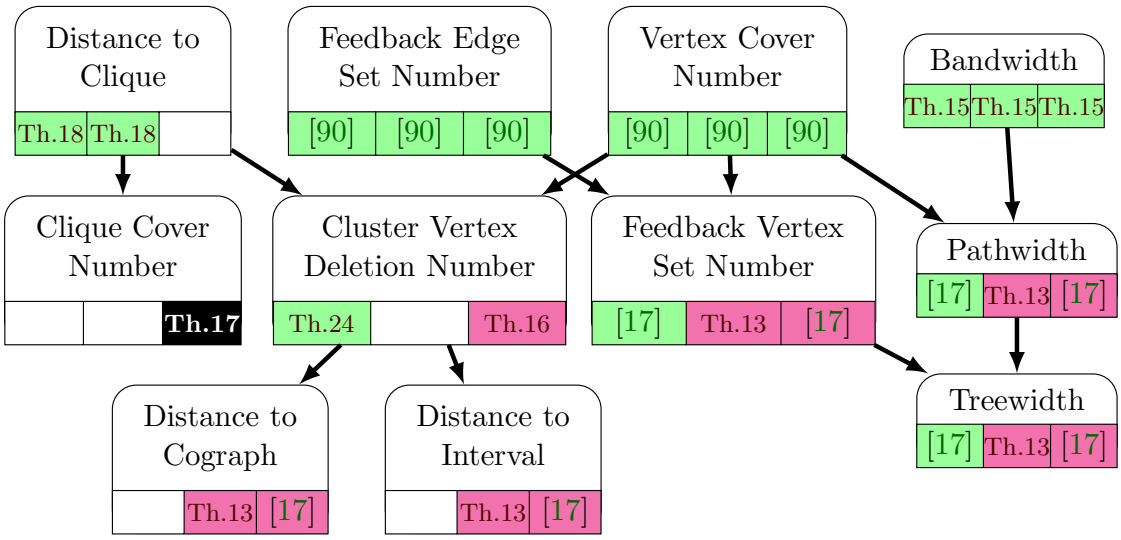


Figure 3.1: Overview of our results for Target Set Selection when parameterized by structural parameters (see Section 2.3.4). The three rectangles below each parameter indicate a result with (from left to right:) constant, majority, and general threshold function. The white text on black background at the parameter "clique cover number" means NP-hard for constant values of this parameter, violet (dark gray) background in the other parameters means W[1]-hard, green (light gray) background means FPT, and white background indicates an open question.

This chapter is organized as follows. The Section 3.2 is dedicated to the problem definitions as well as some preliminaries. In the Sections 3.3 and 3.4, we study the parameterized complexity of Target Set Selection with respect to parameters related to sparse and dense structures, respectively. In Section 3.5, we turn our attention to the parameterized approximability of Max Open Influence and Max Closed Influence. We also consider the complexity of the decision version Influence. Conclusion and open problems are given in Section 3.6.

As the present work was done in close collaboration with another PhD student, I will specify my main contributions to this chapter. First, I extended the W[1]-reduction proof for Target Set Selection and general thresholds to the majority case (Theorem 13). Secondly, I proved the parameterized inapproximability of Max Open Influence and Max Closed Influence (Theorem 26 and Theorem 27), as well as the polynomial-time inapproximability result for the unanimity case (Theorem 30). Furthermore, I also stated the general result of Lemma 7.

3.2 Problem definitions and preliminaries

Before we define formally the investigated problems, we need to define some notions. Let $G = (V, E)$ be a graph and a threshold function $\text{thr} : V \to \mathbb{N}$. In this work, we consider majority thresholds *i.e.* $\text{thr}(v) = \lceil \frac{\deg(v)}{2} \rceil$ for each $v \in V$, unanimity thresholds *i.e.* $\text{thr}(v) = \deg(v)$ for each $v \in V$, and constant thresholds *i.e.* $\text{thr}(v) \leq c$ for each $v \in V$ and some constant $c > 1$.

We now define the propagation process involved in the studied problems. Initially, all vertices of G are not activated and we select a subset $S \subseteq V$. The propagation unfolds in

discrete steps. At time step 0, only the vertices in S are activated. At time step $t+1$, a vertex v is activated if and only if the number of its activated neighbors at time t is at least $\text{thr}(v)$. We apply the rule iteratively until no more activations are possible. Formally, we define the set of vertices that are activated by S at step i as $\mathcal{A}^i_{G,\text{thr}}(S)$ with

$$\begin{aligned} \mathcal{A}^0_{G,\text{thr}}(S) &= S \text{ and} \\ \mathcal{A}^{i+1}_{G,\text{thr}}(S) &= \mathcal{A}^i_{G,\text{thr}}(S) \cup \{v \in V : |N(v) \cap \mathcal{A}^i_{G,\text{thr}}(S)| \geq \text{thr}(v)\} \end{aligned}$$

We denote by $r(S) = \max\{i : \mathcal{A}^{i-1}_{G,\text{thr}} \neq \mathcal{A}^i_{G,\text{thr}}\}$ the number of activation steps before the propagation stops. Notice that $r(S) \leq |V|$ since at least one vertex is activated at each step until the propagation process stops.

Definition 32: Closed activated vertices

Let $G = (V, E)$ be a graph, a threshold function $\text{thr} : V \to \mathbb{N}$, and a set $S \subseteq V$.
The closed activated vertices, denoted by $\sigma_{G,\text{thr}}[S]$, is the set of **all** activated vertices at the end of the propagation process provided that S is the set of initially activated vertices *i.e.* $\sigma_{G,\text{thr}}[S] = \mathcal{A}^{r(S)}_{G,\text{thr}}(S)$.

Definition 33: Open activated vertices

Let $G = (V, E)$ be a graph, a threshold function $\text{thr} : V \to \mathbb{N}$, and a set $S \subseteq V$.
The open activated vertices, denoted by $\sigma_{G,\text{thr}}(S)$, is the set of **newly** activated vertices at the end of the propagation process provided that S is the set of initially activated vertices *i.e.* $\sigma_{G,\text{thr}}(S) = \mathcal{A}^{r(S)}_{G,\text{thr}}(S) \setminus S$.

Definition 34: Target set

Let $G = (V, E)$ be a graph and a threshold function $\text{thr} : V \to \mathbb{N}$, and a set $S \subseteq V$.
We say that S is a target set for (G, thr) if $\sigma_{G,\text{thr}}[S] = V$.

We omit the subscript (G, thr) if the graph and the threshold function are clear from the context. We denote the maximum threshold of an instance (G, thr) by $\text{thr}_{\max}(G, \text{thr}) = \max\{\text{thr}(v) : v \in V(G)\}$. We can now formally define the studied problems.

Target Set Selection
Input: A graph $G = (V, E)$, a threshold function $\text{thr} : V \to \mathbb{N}$ and an integer k.
Question: Is there a target set $S \subseteq V$ for (G, thr) such that $|S| \leq k$?

We also consider the complementary problem of Target Set Selection defined as follows.

Influence
Input: A graph $G = (V, E)$, a threshold function $\text{thr} : V \to \mathbb{N}$, and two integers k and ℓ.
Question: Is there a subset $S \subseteq V$, $|S| \leq k$, such that $|\sigma(S)| \geq \ell$?

In this chapter, we investigate the approximability of the corresponding optimization problem. As previously hinting, the fact that we count or not the initially activated

vertices in the solution might change the approximation factor. Thus we need to define two maximization versions of Influence.

Max Open Influence
Input: A graph $G = (V, E)$, a threshold function $\text{thr} : V \to \mathbb{N}$, and an integer k.
Output: A subset $S \subseteq V$, $|S| \leq k$, such that $|\sigma(S)|$ is maximum.

Max Closed Influence
Input: A graph $G = (V, E)$, a threshold function $\text{thr} : V \to \mathbb{N}$, and an integer k.
Output: A subset $S \subseteq V$, $|S| \leq k$, such that $|\sigma[S]|$ is maximum.

We will sometimes make use of the following relation between Max Open Influence and Max Closed Influence.

Lemma 10 *If there is an α-approximation algorithm for* Max Open Influence *then there is an α-approximation algorithm for* Max Closed Influence.

Proof. Let A be an α-approximation algorithm for Max Open Influence. Let $I = (G, \text{thr}, k)$ be an instance of Max Closed Influence with $|V| = n$ and optimum value $\text{opt}(I)$. When we apply A on I, we obtain a solution S such that $|\sigma(S)| \geq \frac{\text{opt}(I)-k}{\alpha(n)}$ and thus $|\sigma[S]| = k + |\sigma(S)| \geq \frac{\text{opt}(I)}{\alpha(n)}$. ■

We use the following two data reduction rules throughout this work. If the threshold of a vertex exceeds its degree, it cannot be activated by its neighbors and, hence, the vertex is part of any target set. Moreover, we consider vertices with threshold 0 as already active.

Reduction rule 11 ([90, Reduction Rule 1]) *Let $G = (V, E)$ and $v \in V$. If $\text{thr}(v) > \deg(v)$, then delete v, decrease the threshold of all its neighbors by one and decrease k by one. If $\text{thr}(v) = 0$, then delete v and decrease the thresholds of all its neighbors by one.*

In an instance that is reduced with respect to Reduction rule 11, every degree-one vertex has threshold one. Thus, considering an arbitrary degree-one vertex, we do not select it into the target set as choosing its neighbor is at least as good. This is formalized in the next data reduction rule.

Reduction rule 12 ([90, Reduction Rule 5]) *Let (G, thr, k) be an instance of* Target Set Selection *reduced with respect to Reduction rule 11 and let $v \in V(G)$ with $\text{thr}(v) = \deg(v) = 1$. Then, delete v from G.*

Parameter Identification. In this paragraph, we further justify the choice of the structural parameters in this study. Fixed-parameter algorithms are efficient in practice if the considered parameter is small. As previously hinted, target set selection having many applications on social networks [52], it is natural to extract small parameters from typical properties of social networks. A widely accepted property of social networks is the so-called "small-world phenomenom", roughly stating that the diameter of social networks is usually small. Unfortunately, the diameter of the input graph turns out not to be a suitable parameter as the problem is NP-hard for constant diameter values [90]. When the network models friendships, we expect the network to be made up of multiple cliques (or dense substructures) that overlap. This motivates considering the number of cliques

needed to cover all vertices [71] (the "clique cover number") or the number of vertices to remove to obtain a clique (the "distance to clique"). As the latter parameter is somewhat restrictive, we also considered the number of vertices to delete in order to obtain a collection of disjoint cliques (the "cluster vertex deletion number"). Recently, the cluster vertex deletion number was also used to parameterize problems related to coloring and hamiltonicity [48]. In some applications, we deal with very sparse social networks, for instance networks modeling romantic relationship [52, Chap. 2, Fig. 2.7]. In these cases, parameters related to the sparseness of the input graph are interesting. Among them, we consider the number of vertices to remove to obtain an edgeless graph ("vertex cover number"), the number of edges or vertices to remove to obtain a forest ("feedback edge set number" and "feedback vertex set number") as well as some graph width parameters (treewidth, pathwidth, and bandwidth).

3.2.1 Basic reductions I & II

In the following, we present several "basic reductions" that are used as starting point for several reductions in this chapter.

Basic reduction I. This fpt-reduction, which we will refer to as *basic reduction I*, was introduced by Ben-Zwi et al. [17] and is from the W[1]-hard problem Multicolored Clique (see Appendix A). Let (G, col, k) be an instance of Multicolored Clique. An equivalent instance (G', thr, k') of Target Set Selection is constructed as follows. For each color $c \in \{1, \ldots, k\}$, create a *vertex-selection gadget* X_c consisting of a star whose leaves one-to-one correspond to vertices with color c in G. For each pair of distinct colors $c_1, c_2 \in \{1, \ldots, k\}$, let $E_{\{c_1,c_2\}} \subseteq E$ be the set of all edges that connect vertices of color c_1 with vertices of color c_2 and create the following *edge-selection gadget* $X_{\{c_1,c_2\}}$. The edge-selection gadget $X_{\{c_1,c_2\}}$ consists of a star whose leaves one-to-one correspond to edges in $E_{\{c_1,c_2\}}$. The center vertex of any star is called *guard*.

The second type of gadgets is a *validation gadget*. They use the arbitrary bijection $\mathrm{low} : V \to \{1, \ldots, n\}$ and the bijection $\mathrm{high} : V \to \{n, \ldots, 2n-1\}$ defined as $\mathrm{high}(v) = 2n - \mathrm{low}(v)$ for each $v \in V$. For each $\{c_1, c_2\}$ with $c_1, c_2 \in \{1, \ldots, k\}$, add two validation gadgets V_{c_1,c_2} and V_{c_2,c_1} each consisting of two vertices. Now, for each $uv \in E_{\{c_1,c_2\}}$ such that $\mathrm{col}(v) = c_1$, connect the first validation gadget V_{c_1,c_2} as follows: Let v' be the vertex in X_{c_1} corresponding to v. First, add $\mathrm{low}(v)$ vertices and connect them to v' and to the first vertex of V_{c_1,c_2}. Next, add $\mathrm{high}(v)$ vertices and connect them to v' and to the second vertex of V_{c_1,c_2}. Analogously, with e_{uv} denoting the vertex in $X_{\{c_1,c_2\}}$ corresponding to the edge uv, add $\mathrm{high}(v)$ vertices and connect them to e_{uv} and to the first vertex of V_{c_1,c_2}. Then, add $\mathrm{low}(v)$ vertices and connect them to e_{uv} and to the second vertex of V_{c_1,c_2}. The second validation gadget V_{c_2,c_1} is analogously connected to the vertex of X_{c_2} that corresponds to u and to e_{uv} in $X_{\{c_1,c_2\}}$.

We call all the vertices adjacent to vertices of a validation gadget *connection vertices*. The thresholds are set as follows: Guard vertices and connection vertices have threshold one, the two vertices in each validation gadget have threshold $2n$, and the remaining vertices in the selection gadgets have a threshold equal to their degree. Finally, $k' = k + \binom{k}{2}$. This completes the reduction.

As to the correctness: If the instance (G, col, k) is a yes-instance of Multicolored Clique then the vertices chosen to be in the target set of G' refer to the multicolored clique in G: For each vertex in the clique, the corresponding vertex in the vertex-selection gadget is in the target set. Furthermore, for each edge in the clique the corresponding vertex in the edge-selection gadget is in the target set. This target set activates the whole

graph. In the reverse direction the validation gadgets play a central role: Each validation gadget connects a vertex-selection gadget with and edge-selection gadget. The vertices in the validation gadget only become activated if a vertex in the vertex-selection gadget and a vertex in the edge-selection gadget are in the target set such that the corresponding vertex and edge in G are incident. Basically this ensures that one has to choose vertices in G' into the target set that refer to a multicolored clique in G. We refer the reader to Ben-Zwi et al. [17] for more details.

Basic reduction II. In the following, we introduce a fpt-reduction from the W[2]-hard problem Dominating Set (see Appendix A), denoted as *basic reduction II*. Given an instance $(G = (V, E), k)$ of Dominating Set we construct the instance $(G' = (V', E'), \mathrm{thr}, k)$ of Target Set Selection as follows. For each vertex $v \in V$, we add two vertices v^t and v^b (t and b respectively standing for *top* and *bottom*) to V' as well as the edge $v^t v^b$ to E'. For each edge $uv \in E$, add the edges $u^t v^b$ and $u^b v^t$ in G'. Finally, set $\mathrm{thr}(v^t) = \deg_{G'}(v^t)$ and $\mathrm{thr}(v^b) = 1$ for every top vertex v^t and every bottom vertex v^b, respectively. This completes the reduction (see Figure 3.2).

As to the correctness: For the forward direction, suppose there exists a dominating set $S \subseteq V$ in G of size k. Consider the solution $S' \subseteq V'$ containing the corresponding top vertices. After the first step, all bottom vertices are activated since they have thresholds one and S is a dominating set. Finally, after the second step, all top vertices are activated too. For the reverse direction, suppose there is a subset $S' \subseteq V'$ of size k in G' such that $\sigma[S'] = V'$. We can assume without loss of generality that S' contains no bottom vertex. Since all bottom vertices are activated we have that $\{v_i : v_i^t \in S'\}$ is a dominating set in G.

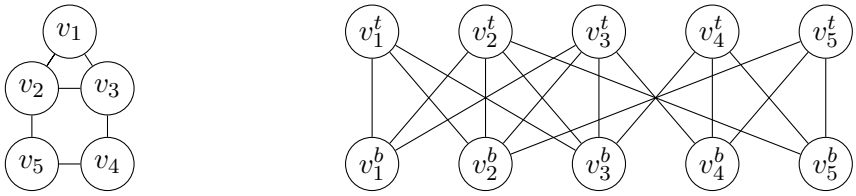


Figure 3.2: Sample construction of the bipartite graph G' (right) from a graph G (left) of Dominating Set. All vertices v_i^t, $1 \leq i \leq 5$ have thresholds $\deg_{G'}(v_i^t)$ while all vertices v_i^b, $1 \leq i \leq 5$ have thresholds 1.

3.3 Parameters related to sparse structures

In this section, we consider parameters that measure the sparseness of the input graph. Since trees are the most sparse connected graphs and Target Set Selection is polynomial-time solvable on trees [35], parameters measuring the distance to trees are most interesting. Canonical candidates for this are the treewidth, the pathwidth, and the feedback vertex set number of the input graph. Notably, if the maximum threshold $\mathrm{thr}_{\max}$ is bounded by a constant, then a fixed-parameter algorithm of Ben-Zwi et al. [17] for the parameter treewidth tw can solve Target Set Selection in $\mathrm{thr}_{\max}^{O(\mathrm{tw})} \cdot n^{O(1)}$ time, implying fixed-parameter tractability for the three parameters mentioned above. Furthermore, Ben-Zwi et al. [17] proved W[1]-hardness for Target Set Selection with respect to the parameter treewidth when the thresholds are unbounded. We extend this result by showing W[1]-hardness for treewidth when the thresholds respect the majority condition. The

proof even shows hardness for the combined parameter feedback vertex set, pathwidth, distance to cographs, and distance to interval graphs. Finally, we show that Target Set Selection is fixed-parameter tractable when parameterized by the bandwidth. This result even holds for general thresholds.

3.3.1 Extending the basic reduction

In the following, we show that the basic reduction I (see Section 3.2.1) can be extended to prove hardness for the combination of various sparseness-related parameters.

Theorem 13 Target Set Selection *with majority threshold is* W[1]*-hard with respect to the combination of the following parameters: feedback vertex set, distance to cograph, distance to interval graph, and pathwidth.*

Proof. We modify the basic reduction I to get the new, equivalent instance $(G'', \mathrm{thr}', k'')$ as follows. For each vertex v in a validation gadget, add $\deg_{G'}(v)-4n$ vertices adjacent to v. Moreover, for each guard vertex v add $\deg_{G'}(v) - 2$ neighbors. Let X be the set of vertices added so far. Insert a new vertex u adjacent to all vertices in X and add $|X|+2(k'+2)$ vertices to the neighborhood of u. To complete the modification of the graph, for every vertex v in a selection gadget, attach $\deg_{G'}(v)$ neighbors. Finally, set $\mathrm{thr}'(v) = \lceil \deg_{G''}(v)/2 \rceil$ for all $v \in V(G'')$ and $k'' = k' + 1$.

We claim that $(G'', \mathrm{thr}', k'')$ is a yes-instance if and only if (G', thr, k') is a yes-instance.

"$\Rightarrow$": Suppose that there is a solution S' for $(G'', \mathrm{thr}', k'')$. First, observe that $u \in S'$ since, otherwise, u would not become active. Indeed, even if all the vertices in G'' plus k'' degree-one neighbors of u are activated, the vertex u will not be activated since its threshold is $|X| + k'' + 1$. Since u is in all solutions, we may consider the equivalent instance where u is removed together with all its neighbors (they all have threshold one and thus get activated by u). Moreover, for each removed vertex v, we have to decrease the threshold of the vertices in $N(v)$ by one. This operation leaves a graph with many degree-one vertices of threshold one. Applying Reduction rule 12, we arrive at the instance (G', thr, k'). By the correctness of Reduction rule 12, it follows that S' is also a solution for (G', thr, k').

"$\Leftarrow$": Conversely, let S be a solution for (G', thr, k'). Since activating u and exhaustively applying Reduction rule 12 on $(G'', \mathrm{thr}', k'')$ results in (G', thr, k'), it is clear that $S \cup \{u\}$ is a target set for (G'', thr') of size $k' + 1$.

To complete the proof of the theorem, it is enough to observe that if we remove the vertex u, all guard vertices, and the validation gadgets (that is, $\binom{k}{2} + k + 1$ vertices), then we get stars and isolated vertices. ■

3.3.2 Bandwidth

Another possible measure for sparseness is the bandwidth of the input graph. Here, our result is of more positive nature: we show that Target Set Selection is fixed-parameter tractable with respect to the bandwidth, even for general threshold functions, by using an algorithm of Ben-Zwi et al. [17].

Theorem 14 Target Set Selection *is fixed-parameter tractable with respect to the combined parameter "treewidth" and "maximum degree" of the input graph.*

Proof. Let $(G = (V, E), \mathrm{thr}, k)$ be an instance of Target Set Selection. First, exhaustively apply Reduction rule 11 to get a new equivalent instance $(G' = (V', E'), \mathrm{thr}', k')$. Observe that $\mathrm{thr}'(v) \leq \deg_{G'}(v)$ for all $v \in V'$. Moreover, Ben-Zwi et al. [17] gave a $(\mathrm{thr}_{\max})^{O(\mathrm{tw})} \cdot n$-time algorithm for solving Target Set Selection, where tw is the treewidth of the input graph and $\mathrm{thr}_{\max}$ is the maximum threshold value. It follows that this algorithm applied to G' runs in time $\Delta(G')^{O(\mathrm{bw}(G'))} \cdot n$ since $\mathrm{tw}(G') \leq 2\,\mathrm{bw}(G')$. ■

By the definition of bandwidth we can deduce that $\Delta(G) \leq 2\,\mathrm{bw}(G)$ for any graph G which implies the following corollary.

Corollary 15 Target Set Selection *is fixed-parameter tractable with respect to the parameter "bandwidth" of the input graph.*

3.4 Parameters related to dense structures

In contrast to the previous section, we now consider Target Set Selection with respect to parameters related to the denseness of the input graph. Since cliques are the most dense graphs and Target Set Selection is polynomial-time solvable on cliques [90], parameters measuring the distance to cliques are most interesting. In particular, we consider the vertex deletion distance to clique and to a collection of disjoint cliques (also called cluster vertex deletion number or "cvd number" for short), and the clique cover number.

Starting with the case of unrestricted thresholds in Section 3.4.1, we show that Target Set Selection parameterized by the size of a minimum cluster vertex deletion (cvd) set is W[1]-hard. Furthermore, we show NP-hardness when restricting to instances with clique cover number two. Then, in Section 3.4.2, we study restricted threshold functions. For constant or majority thresholds, Target Set Selection parameterized by the distance to a clique is fixed-parameter tractable. Furthermore, we show an exponential-size kernel with respect to the combined parameter maximum threshold value and cvd number, implying fixed-parameter tractability with respect to the cvd number on inputs with thresholds bounded by a constant.

3.4.1 Unrestricted thresholds

Here, we research the general Target Set Selection setting without constraints on the thresholds of the input. As hinted in the introduction, the next two theorems state that these variants are parameterized intractable with respect to the employed denseness measures.

Theorem 16 Target Set Selection *is* W[1]*-hard with respect to the parameter "cvd number".*

Proof. We modify the basic reduction I (see Section 3.2.1) as follows. Make all connection vertices that have a common vertex in a (vertex- or edge-) selection gadget to a clique. The thresholds of the connection vertices are modified as follows: In each maximal clique of the connection vertices, an arbitrary ordering of the vertices is fixed. Then the first vertex has threshold one, the second has threshold two, and the i^{th} vertex has threshold i. See Figure 3.3 for a scheme of the reduction.

Note that each connection vertex is contained in exactly one maximal clique.

Notice also that the following holds. Let C be such a maximal clique, V^1 (resp. V^1 and V^2) denote the validation gadget adjacent to C, and v the vertex adjacent to C in the vertex-selection gadget (resp. the edge-selection gadget). Then all connection vertices in C are activated if and only if either v is activated or all the vertices in V^1 (*resp.* V^1 and V^2) are activated.

We now prove that (G, col, k) is a yes-instance of Multicolored Clique if and only if $(G', \mathrm{thr}, \binom{k}{2} + k)$ is a yes-instance of Target Set Selection.

"⇒": Suppose that (G, col, k) has a multicolored clique $C \subseteq V$ of size k. Then the set $S = \{v \in C\} \cup \{e_{uv} : u, v \in C\}$ is a target set for (G', thr, k'). Indeed, in the first step of the propagation process every guard vertex is activated since they are all adjacent to a vertex in S. After $4n$ steps, all the connection vertices adjacent to a vertex in S get activated. During the next step, all $4\binom{k}{2}$ vertices in validation pairs will be activated since C is a multicolored clique of size k. From now on, it is not hard to see that the entire graph will be activated.

"⇐": Conversely, assume that (G', thr, k') has a target set $S \subseteq V'$ of size k. First, we may assume that S does not contain any guard vertex since they all have threshold one. Moreover, one has to pick up in the target set at least one vertex in each selection gadget to activate the guard vertex of the latter. Indeed, recall that every neighbor of a guard vertex has a threshold equal to its degree and the guard vertex is not in the target set. Thus, every target set contains at least one vertex in each selection gadget. Furthermore, since $k' = \binom{k}{2} + k$ we conclude that there is *exactly* one vertex from each selection gadget in a minimal target set. Suppose now that we select two vertices $u \in X_{c_1}$ and $v \in X_{c_2}$ together with an edge-vertex $e_{u'v'} \in X_{\{c_1, c_2\}}$ for some $c_1, c_2 \in \{1, \ldots, k\}$ such that $e_{u'v'}$ is not incident to both u and v. Without loss of generality, we may assume that $u \neq u'$. Then at least one vertex in the validation gadgets V_{c_1,c_2} and V_{c_2,c_1} will not be activated. To see this, recall that for all $w \in V'$ it holds that $\mathrm{high}(w) + \mathrm{low}(w) = 2n$ and, since $u \neq u'$, we have either $\mathrm{low}(u) + \mathrm{high}(u') < 2n$ or $\mathrm{high}(u) + \mathrm{low}(u') < 2n$. This implies, as previously discussed, that some connection vertices will not be activated, a contradiction. ■

Theorem 17 Target Set Selection *is* NP*-hard and* W[2]*-hard with respect to the parameter k even on graphs with clique cover number two.*

Proof. We present a parameterized reduction from the W[2]-hard problem Hitting Set (see Appendix A). Given an instance $(\mathcal{F}, U, k)$ of Hitting Set consisting of a set family $\mathcal{F} = \{F_1, \ldots, F_m\}$ over a universe $U = \{u_1, \ldots, u_n\}$ and an integer $k \geq 0$, we construct an instance (G, thr, k) of Target Set Selection as follows.

We start with the construction of the graph G. A set of vertices V_U contains a vertex for every element $u \in U$, that is, $V_U = \{v_u : u \in U\}$. Analogously, a second set $W_\mathcal{F}$ contains a vertex for every subset, that is, $W_\mathcal{F} = \{w_F : F \in \mathcal{F}\}$. The vertices in V_U are called *element vertices* and the vertices in $W_\mathcal{F}$ are called *subset vertices*. There is an edge between an element vertex v_u and a subset vertex w_F if and only if $u \in F$. Next, add a new vertex $x \notin (V_U \cup W_\mathcal{F})$ to G and connect x to all vertices in $W_\mathcal{F}$. Then, make $V_1 = V_U \cup \{x\}$ a clique. Add $|\mathcal{F}| - 1$ sets of vertices $V_1^B, \ldots, V_{|\mathcal{F}|-1}^B$ to the graph, each set containing $\alpha = |U| + 2$ vertices and let $V^B = \bigcup_{i=1}^{|\mathcal{F}|-1} V_i^B$. Finally, make $V_2 = W_\mathcal{F} \cup V_1^B \cup \ldots \cup V_{|\mathcal{F}|-1}^B$ a clique.

The thresholds are set as follows. For every subset vertex $w_{F_i} \in W_\mathcal{F}$, set the

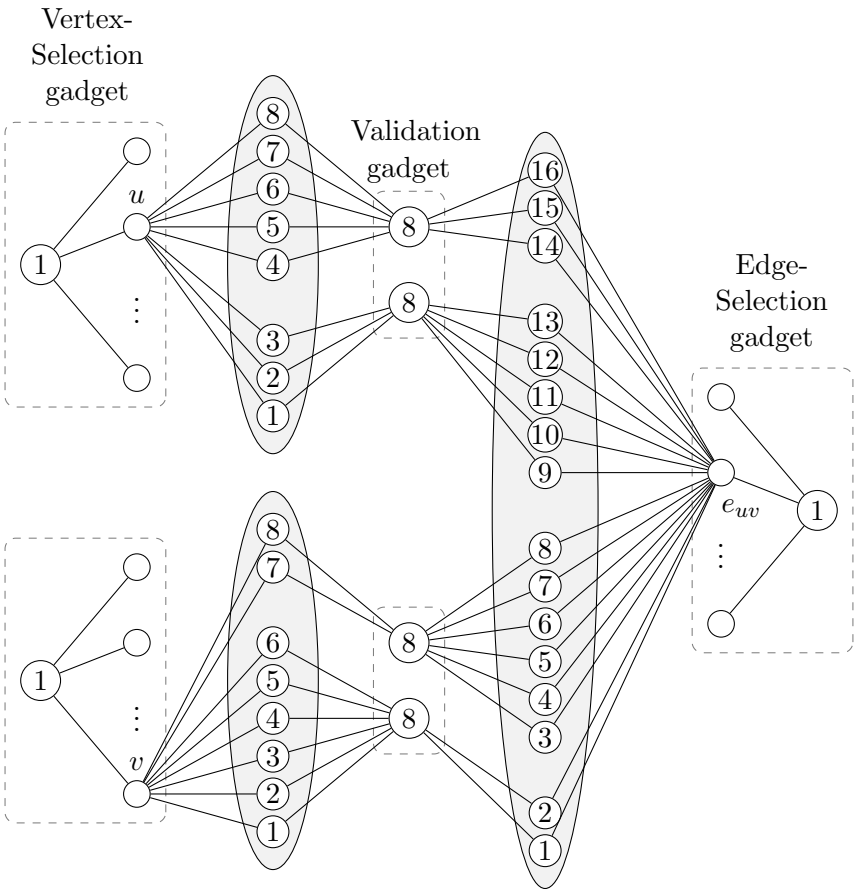


Figure 3.3: Graph obtained after carrying out the modifications in the proof of Theorem 16 (with $n = 4$). The vertices inside an ellipse form a clique. The numbers in the vertices denote the thresholds. If no number is inside a vertex, the threshold is equal to the degree.

threshold $\text{thr}(w_{F_i}) = (i-1)\alpha+i$, for every element vertex $v_u \in V_U$, set $\text{thr}(v_u) = |\{F \in \mathcal{F} : u \in F\}|+k+1$, and for each vertex $v \in V_i^B$, $1 \leq i \leq |\mathcal{F}|-1$, set $\text{thr}(v) = (i-1)\alpha+i$. Finally, complete the construction by setting $\text{thr}(x) = |W_{\mathcal{F}}| + k$

Since V_1 and V_2 are cliques, the constructed graph G is a diameter-two graph whose vertices can be covered by two cliques, see Figure 3.4.

For the correctness it remains to show that $(\mathcal{F}, U, k)$ is a yes-instance of Hitting Set if and only if (G, thr, k) is a yes-instance of Target Set Selection.

"⇒": If $(\mathcal{F}, U, k)$ is a yes-instance, then there exists a hitting set U' of size k for $\mathcal{F}$. We show that $S = \{v_u : u \in U'\}$ is a target set for G of size k . Since U' is a hitting set, every vertex in $W_{\mathcal{F}}$ has at least one neighbor in S. Thus, all vertices in V_1 become active in $2|W_{\mathcal{F}}| - 1$ steps: In the first step w_{F_1} is activated since $\text{thr}(w_{F_1}) = 1$. Then in the second step, all vertices in V_1^B are activated since all these vertices also have threshold one and w_{F_1} is active. For $2 \leq i \leq |W_{\mathcal{F}}|$, in the $(2i-1)^{\text{th}}$ step the vertex w_{F_i} is activated and in the next step all vertices in V_i^B: The neighbors of w_{F_i} that are active in step $2i-2$ are: All vertices in $V_1^B \cup \ldots \cup V_{i-1}^B$, the vertices $w_{F_1}, \ldots, w_{F_{i-1}}$ and at least one vertex in S. Since the threshold is $\text{thr}(w_{F_i}) = (i-1)\alpha + i$, the vertex w_{F_i} is activated. Then, there are $(i-1)\alpha + i$ active vertices in V_2 and, hence, all vertices in V_i^B are activated in the $2i^{\text{th}}$ step. After all vertices in V_2 are active, x

is activated. Finally, in the last step all vertices in $V_U \setminus S$ are activated since for every vertex in $V_U \setminus S$ all neighbors in $W_{\mathcal{F}}$ and x have been activated.

"$\Leftarrow$": If (G, thr, k) is a yes-instance of Target Set Selection, then there is a target set S of size at most k. We first show that $S \subseteq V_U$.

Assume towards a contradiction that there is a vertex in $S \setminus V_U$. Then, $|S \cap V_U| \leq k-1$. Let ℓ denote the first step in which a vertex in $V_U \setminus S$ is activated, that is, $\ell = \min\{j : \mathcal{A}^j_{G,\mathrm{thr}}(S) \cap (V_U \setminus S) \neq \emptyset\}$. Moreover, let $v_u \in \mathcal{A}^\ell_{G,\mathrm{thr}}(S) \cap (V_U \setminus S)$. Note that, by definition of ℓ, it holds that $|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V_1| \leq k$. Hence

$$\begin{aligned}|N_G(v_u) \cap \mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S)| =& |N_G(v_u) \cap \mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap W_{\mathcal{F}}| \\ &+ |N_G(v_u) \cap \mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V_1| \\ \leq& |\{F \in \mathcal{F} : u \in F\}| + k \\ <& |\{F \in \mathcal{F} : u \in F\}| + k + 1 = \mathrm{thr}(v_u)\end{aligned}$$

a contradiction. Therefore, $S \subseteq V_U$.

Finally, we show that $U' = \{u : v_u \in S\}$ is a hitting set for $\mathcal{F}$. To this end, we show that every subset vertex w_{F_i} has a neighbor in S and, hence, is hit by U'. Assume towards a contradiction that there exists a vertex $w_{F_i} \in W_{\mathcal{F}}$ with $N_G(w_{F_i}) \cap S = \emptyset$. Let $X^i = (\bigcup_{i \leq j \leq |\mathcal{F}|-1} V^B_j) \cup (\bigcup_{i \leq j \leq |\mathcal{F}|} \{w_{F_i}\})$. Let ℓ denote the first step in which a vertex in X^i is activated, that is, $\ell = \min\{j : \mathcal{A}^j_{G,\mathrm{thr}}(S) \cap X^i \neq \emptyset\}$. Hence $|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap W_{\mathcal{F}}| = |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap (W_{\mathcal{F}} \setminus X^i)| \leq i-1$ and $|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V^B| = |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap (V^B \setminus X^i)| \leq |\bigcup_{1 \leq j < i} V^B_j| = (i-1)\alpha$. Let $v \in X^i \cap V^B$, then we have

$$\begin{aligned}|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap N_G(v)| &= |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V^B| + |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap W_{\mathcal{F}}| \\ &\leq (i-1)\alpha + i - 1 \\ &< (i-1)\alpha + i \leq \mathrm{thr}(v)\end{aligned}$$

and, thus, $\mathcal{A}^\ell_{G,\mathrm{thr}}(S) \cap X^i \cap V^B = \emptyset$. Now consider $v \in X^i \cap (W_{\mathcal{F}} \setminus \{w_{F_i}\})$. Observe that $|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}} \cap V_1| = |S| = k$. Thus, we have

$$\begin{aligned}|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap N_G(v)| &\leq |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V^B| + |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap W_{\mathcal{F}}| + |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V_1| \\ &\leq (i-1)\alpha + i - 1 + k \\ &< i\alpha + i + 1 \leq \mathrm{thr}(v)\end{aligned}$$

and, thus, $\mathcal{A}^\ell_{G,\mathrm{thr}}(S) \cap X^i \cap (W_{\mathcal{F}} \setminus \{w_{F_i}\}) = \emptyset$. Finally, consider $v = w_{F_i}$

$$\begin{aligned}|\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap N_G(v)| &\leq |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap V^B| + |\mathcal{A}^{\ell-1}_{G,\mathrm{thr}}(S) \cap W_{\mathcal{F}}| \\ &\leq (i-1)\alpha + i - 1 \\ &< (i-1)\alpha + i = \mathrm{thr}(v)\end{aligned}$$

and, thus, $w_{F_i} \notin \mathcal{A}^\ell_{G,\mathrm{thr}}(S)$. Altogether we have $\mathcal{A}^\ell_{G,\mathrm{thr}}(S) \cap X^i \cap V^B = \emptyset$, $\mathcal{A}^\ell_{G,\mathrm{thr}}(S) \cap X^i \cap (W_{\mathcal{F}} \setminus \{w_{F_i}\}) = \emptyset$, and $w_{F_i} \notin \mathcal{A}^\ell_{G,\mathrm{thr}}(S)$ and, hence, $\mathcal{A}^\ell_{G,\mathrm{thr}}(S) \cap X^i = \emptyset$, a contradiction. ■

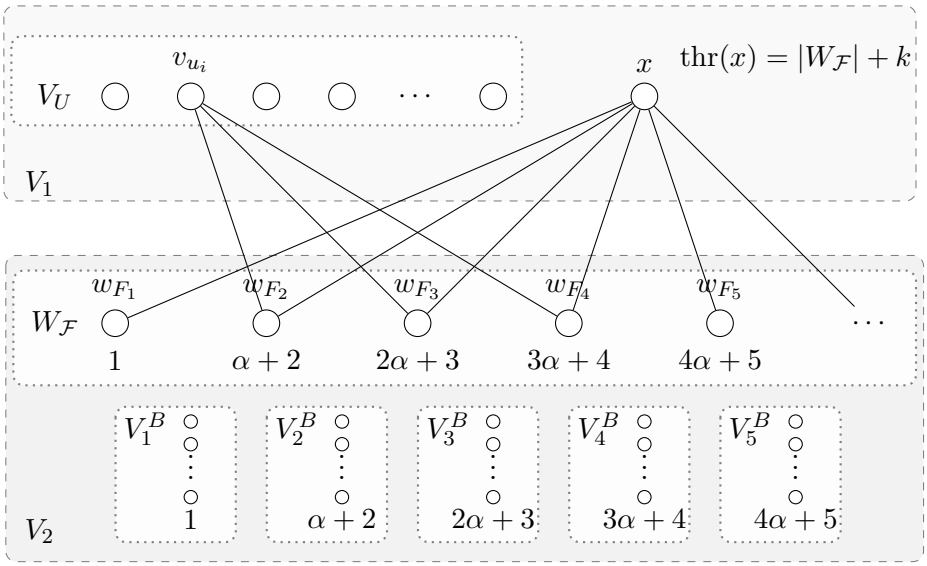


Figure 3.4: A schematic picture of the constructed graph. Each of the two vertex sets V_1 (upper box) and V_2 (lower box) form a clique. The number below a vertex denotes its threshold. The only way to activate all vertices with a target set of size k is to choose k vertices in V_U such that these k vertices activate all vertices in $W_{\mathcal{F}}$.

3.4.2 Restricted thresholds

In the spirit of researching the influence of bounded thresholds on Target Set Selection, we consider the parameters distance to clique and cluster vertex deletion number (cvd number). Recall that we showed W[1]-hardness for the parameter cvd number (for unbounded thresholds) in the previous paragraph. By presenting an exponential-size kernel, we show that the problem becomes tractable with respect to this parameter if the maximum threshold is a constant.

First, we show that Target Set Selection with majority thresholds or constant thresholds is fixed-parameter tractable with respect to the parameter distance to clique. We can even show fixed-parameter tractability for less restrictive threshold functions. To this end, let $\mathcal{P}(V)$ be the set of all subsets of V.

Theorem 18 Target Set Selection *on graphs with vertex set V is fixed-parameter tractable with respect to the parameter "distance to clique", denoted by ℓ, when the threshold function* thr *fulfills the restriction*

$$\text{thr}(v) > g(\ell) \Rightarrow \text{thr}(v) = f(N(v))$$

for all vertices $v \in V$ and arbitrary functions $f : \mathcal{P}(V) \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$.

Proof. We prove the theorem by giving a fixed-parameter algorithm computing a target set of minimum size for (G, thr). To this end, we introduce some notations. Let $X \subset V$, $|X| = \ell$, denote a set of vertices such that $G[V \setminus X]$ is a clique. We define a non-standard "twins" equivalence relation $\equiv$ by

$$u \equiv v \iff (N[u] = N[v]) \wedge (\text{thr}(u) = \text{thr}(v)) \wedge (u \in X \iff v \in X)$$

Since the thresholds and neighborhoods of all vertices in an equivalence class Z are equal, we can denote this threshold and this neighborhood by $\mathrm{thr}(Z)$ and $N[Z]$, respectively. Let $Z_1, Z_2, \ldots, Z_s$ be a list of all nonempty equivalence classes of $\equiv$. Since $G[V \setminus X]$ is a clique, we know that for all $u, v \in V \setminus X : N[u] = N[v]$ if and only if $N_{G[X]}[u] = N_{G[X]}[v]$. Due to the condition $\mathrm{thr}(v) > g(\ell) \Rightarrow \mathrm{thr}(v) = f(N(v))$, for each subset $X' \subseteq X$, there are at most $g(\ell) + 1$ equivalence classes disjoint from X whose neighborhood in X is exactly X'. Hence, $s \leq 2^{\ell}(g(\ell) + 1) + \ell$.

Let S be a minimum-size target set for (G, thr). With S, we can define r_i as the number of the first activation step in which all vertices of Z_i are active. More formally, $r_i = \min\{j : Z_i \subseteq \mathcal{A}^j_{G,\mathrm{thr}}(S)\}$. Let $r = \max\{r_i : 1 \leq i \leq s\}$.

In the following, we upper-bound r by s. We do this by showing that for each $1 \leq j \leq r$, there is an $1 \leq i \leq s$ such that $r_i = j$. Assume this was false, that is, there is some activation step j such that none of the equivalence classes gets activated in step j. Since $j \leq r$, there is some vertex v that gets activated in step j. Let Z_i denote the equivalence class of v. Since $j \geq 1$, we know that $|N(v) \cap \mathcal{A}^{j-1}_{G,\mathrm{thr}}(S)| \geq \mathrm{thr}(v)$. Since for each vertex $u \in Z_i$, $\mathrm{thr}(u) = \mathrm{thr}(v)$ and $N(u) = N(v)$, we conclude that $Z_i \subseteq \mathcal{A}^j_{G,\mathrm{thr}}(S)$, contradicting the assumption that $r_i \neq j$.

Now we describe our algorithm. In the first phase, we guess the correct values of r_i for all $1 \leq i \leq s$. There are at most $r^s \leq s^s$ possibilities to do so.

In the second phase of the algorithm, we use an ILP formulation to solve the problem. Each variable x_i in the ILP represents the number of vertices in the equivalence class i that are in the target set S. We use constraints to model the activation process: For each equivalence class Z_i, the number of active neighbors in step r_i exceeds $\mathrm{thr}(Z_i)$. Two types of active neighbors are considered. First, the vertices in $N[Z_i] \cap S$. Second, the vertices in all equivalence classes $Z_j \subseteq N[Z_i]$ that are active in step i, that is, $r_j < r_i$. More formally,

$$\begin{aligned}
\text{Minimize:} \quad & \sum_{i=1}^{s} x_i \\
\text{subject to:} \quad & \forall 1 \leq i \leq s, \mathrm{thr}(Z_i) \leq \sum_{\substack{Z_j \subseteq N[Z_i] \\ r_j \geq r_i}} x_j + \sum_{\substack{Z_j \subseteq N[Z_i] \\ r_j < r_i}} |Z_j| \\
& \forall 1 \leq i \leq s, x_i \in \{0, 1\}
\end{aligned}$$

By the discussion above, a solution to this ILP corresponds to a minimum-size target set for (G, thr). Since the ILP formulation has s variables, a result by Lenstra [79] implies that solving it is fixed-parameter tractable with respect to s. Since at most s^s such ILPs have to be solved and $s \leq 2^{\ell}(g(\ell)+1)+\ell$, fixed-parameter tractability with respect to ℓ follows. ■

Clearly, Theorem 18 is a pure complexity classification result. Since the majority thresholds and constant thresholds both satisfy the restrictions required in Theorem 18, the next corollary immediately follows.

Corollary 19 Target Set Selection *with majority thresholds or constant thresholds is fixed-parameter tractable with respect to the parameter distance to clique.*

Next, we show fixed-parameter tractability for Target Set Selection with respect to the parameter "cvd number" and constant thresholds. In the following, we assume that an optimal cvd set X of the input graph is given. If this is not the case, then one

might instead use a simple polynomial-time 3-approximation algorithm.[1] Either way, we abbreviate $\ell = |X|$.

In this section we use the notion of "critical cliques". Here, a clique K in a graph is *critical* if all its vertices have the same closed neighborhood and K is maximal with respect to this property.

First, we present a data reduction rule allowing us to bound the number of vertices with the same open or closed neighborhood by the maximum threshold $\text{thr}_{\max}$.

Reduction rule 20 *Let $I = (G = (V, E), \text{thr}, k)$ be an instance of* Target Set Selection *that is reduced with respect to Reduction rule 11 and let $v_1, v_2, \ldots, v_{\text{thr}_{\max}+1} \in V$ be vertices such that either*

$$N(v_1) = N(v_2) = \ldots = N(v_{\text{thr}_{\max}+1}) \text{ or } N[v_1] = N[v_2] = \ldots = N[v_{\text{thr}_{\max}+1}]$$

Furthermore, let v_1 be the vertex with the highest threshold, that is, for all $1 \leq i \leq \text{thr}_{\max}+1$ it holds that $\text{thr}(v_1) \geq \text{thr}(v_i)$. Then delete v_1.

Lemma 21 *Reduction rule 20 is correct and can be applied exhaustively in $O(n+m)$ time.*

Proof. For the running time, note that computing the critical cliques of a graph can be done in linear time [87]. Thus, we first compute the critical cliques of the graph in linear time. Then we iterate over the critical cliques and if one of them has size $\text{thr}_{\max}+r$, $r > 0$, then we delete the r vertices of this critical clique having the largest thresholds. This can clearly be done in linear time. Notice that a maximal set of vertices with the same open neighborhood form a critical clique in the complement graph. Hence, in a second step, we repeat the procedure with the complement graph. Then the graph is reduced with respect to Reduction rule 20. Furthermore observe that Reduction rule 11 is not applicable after Reduction rule 20 was applied.

To show the correctness, we prove that the instance $(G' = (V', E'), \text{thr}, k)$ that is produced by Reduction rule 20 is a yes-instance if and only if the input instance I is a yes-instance.

"$\Rightarrow$:" Since (G', thr, k) is a yes-instance, there exists a target set $S \subseteq V'$, $|S| \leq k$, that activates all vertices in G'. Hence, S activates all vertices of $V \setminus \{v_1\}$ in G. Since (G, thr, k) is reduced with respect to Reduction rule 11, the vertex v_1 is activated by its neighbors. Thus, S is also a target set for (G, thr, k).

"$\Leftarrow$:" Since (G, thr, k) is a yes-instance, there exists a target set $S \subseteq V$, $|S| \leq k$, activating all vertices in G. Let $W = \{v_1, v_2, \ldots, v_{\text{thr}_{\max}+1}\}$ be the vertices considered in the reduction rule. First observe that we can assume $W \setminus S \neq \emptyset$, (that is, not all vertices of W are in the target set) since otherwise, $S' = S \setminus \{v_1\}$ is also a target set: In the first activation step all vertices in $N(v_1)$ become active and, since (G, thr, k) is reduced with respect to Reduction rule 11, it follows that v_1 is active after the second step. Thus, S' is also a target set.

Now consider the case that $v_1 \notin S$. Since for all $v_i \in W$ it holds that $\text{thr}(v_1) \geq \text{thr}(v_i)$ and v_1 is activated by its neighbors, it is clear that all vertices in $W \setminus \{v_1\}$ are active once v_1 is active. Since $|W| > \text{thr}_{\max}$ this implies that all vertices in $N_G(v_1)$ become active in G' and, thus, S is a target set for G'.

[1] A graph is a cluster graph if and only if it contains no induced P_3, that is, an induced path of three vertices. Using this characterization, the factor 3-approximation simply deletes all vertices occurring in an induced P_3.

Finally, consider the case that $v_1 \in S$. Let $w \in W \setminus S$ be the vertex with the highest threshold, that is, for all $v_i \in W \setminus S$ it holds that $\text{thr}(w) \geq \text{thr}(v_i)$. Observe that $S' = (S \setminus \{v_1\}) \cup \{w\}$ is a target set for G': Since S activates all vertices in W it is clear that S' activates all vertices in $W \setminus \{v_1\}$. This implies that all vertices in $N(v_1)$ are activated by S' in G' since $|W \setminus \{v_1\}| = \text{thr}_{\max}$ and all vertices in W have the same neighborhood. Thus, S' activates all vertices in G'. ■

In the following we assume that the input graph G is reduced with respect to Reduction rule 20. Thus, $G[V \setminus X]$ consists of disjoint cliques, each of size at most $2^\ell \, \text{thr}_{\max}$. Hence, in order to get the desired kernel it remains to bound the number of cliques in $G[V \setminus X]$. To this end, we introduce the following notation:

Definition 35: Clusters equivalence

Let $I = (G = (V, E), \text{thr}, k)$ be an instance of Target Set Selection, let $X \subseteq V$ be a cvd set, and let $S \subseteq V$. Let $C_1, C_2 \subseteq V$ be two clusters in $G[V \setminus X]$.
We call C_1 and C_2 *equivalent with respect to* X, denoted by $C_1 \equiv_X C_2$, if there exists a bijection $f : C_1 \to C_2$ such that for every $v \in C_1$ it holds that $\text{thr}(v) = \text{thr}(f(v))$ and $N(v) \cap X = N(f(v)) \cap X$. Furthermore, we call C_1 and C_2 *equivalent with respect to* X *and* S, denoted by $C_1 \equiv_X^S C_2$, if the bijection f additionally fulfills $v \in S \iff f(v) \in S$ for all $v \in C_1$.

Note that $\equiv_X$ is an equivalence relation on the clusters in $G[V \setminus X]$ with at most $(\text{thr}_{\max}+1)^{2^\ell \, \text{thr}_{\max}}$ equivalence classes. To see this, observe that each equivalence class is uniquely determined by 2^ℓ (possibly empty) sequences of thresholds. One for each subset of X. Since G is reduced with respect to Reduction rule 20, each such sequence contains between 0 and $\text{thr}_{\max}$ thresholds. Since each threshold is at most $\text{thr}_{\max}$, the number of equivalence classes is at most

$$\left(\sum_{i=0}^{\text{thr}_{\max}} \text{thr}_{\max}^i \right)^{2^\ell} \leq \left((\text{thr}_{\max}+1)^{\text{thr}_{\max}} \right)^{2^\ell} = (\text{thr}_{\max}+1)^{2^\ell \, \text{thr}_{\max}}$$

In the following, our goal is to bound the number of cliques in one equivalence class in a function depending only on $\text{thr}_{\max}$ and ℓ. Note that once we achieve this goal, we have a kernel with respect to the parameter "cvd number". The next lemma is a first step towards this goal.

Lemma 22 *Let $I = (G = (V, E), \text{thr}, k)$ be an instance of* Target Set Selection, *let $X \subseteq V$ be a cvd set for G, and let $S \subseteq V$, $|S| \leq k$, be a target set for G. Furthermore let $C_1, C_2, \ldots, C_{\text{thr}_{\max}+1} \subseteq V$ be clusters in $G[V \setminus X]$ that are pairwise equivalent with respect to X and S. Then, $S \setminus C_1$ is a target set for $G[V \setminus C_1]$.*

Proof. Let $S' = S \setminus C_1$ and $G' = G[V \setminus C_1]$. We prove the lemma by contradiction: Assume that S' is not a target set for G'. Let $Y \subseteq V \setminus C_1$ be the set of vertices that are activated in G in some step i but are not activated in G' in the step i. Formally, $Y = \{v \in V \setminus C_1 : \exists i \geq 1 \text{ s.t. } v \in \mathcal{A}^i_{G,\text{thr}}(S) \text{ and } v \notin \mathcal{A}^i_{G',\text{thr}}(S')\}$. Since S' is not a target set for G', the set Y is not empty. In particular, Y contains all vertices in G' that are not activated by S'. Let $v \in Y$ be the vertex that is activated first in G *i.e.* for all $u \in Y$ it holds that $u \in \mathcal{A}^i_{G,\text{thr}}(S) \Rightarrow v \in \mathcal{A}^i_{G,\text{thr}}(S)$, $1 \leq i$.

Since $v \in Y$ and $Y \subseteq V \setminus C_1$, it holds that $v \notin S$. Let $i \geq 1$ be the step in which v

becomes active in G, that is, $v \in \mathcal{A}^i_{G,\mathrm{thr}}(S) \setminus \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S)$. Thus, $|N_G(v) \cap \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S)| \geq \mathrm{thr}(v)$. Since v is in G' not activated by S', it follows that $|N_{G'}(v) \cap \mathcal{A}^{i-1}_{G',\mathrm{thr}}(S')| < \mathrm{thr}(v)$. From the selection of v it follows that $Y \cap \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S) = \emptyset$. Thus, $\mathcal{A}^{i-1}_{G,\mathrm{thr}}(S) \setminus \mathcal{A}^{i-1}_{G',\mathrm{thr}}(S') \subseteq C_1$. Since $N_G(v) \setminus N_{G'}(v) \subseteq C_1$, it follows that $N_G(v) \cap \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S) \cap C_1 \neq \emptyset$ and $v \in X$. Let $u \in N_G(v) \cap \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S) \cap C_1$. Note that C_1 and C_j, $1 < j \leq \mathrm{thr}_{\max}+1$, are equivalent with respect to X and S and, hence, there is a bijection f_j as described in Definition 35. Thus, it is easy to see that $u \in \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S) \Rightarrow f_j(u) \in \mathcal{A}^{i-1}_{G,\mathrm{thr}}(S)$. Moreover, since $u \in N_G(v)$ it follows that $f_j(u) \in N_G(v)$ and, thus, $f_j(u) \in N_{G'}(v)$. Hence, $f_j(u) \in N_{G'}(v) \cap \mathcal{A}^{i-1}_{G',\mathrm{thr}}(S')$ for all $2 \leq j \leq \mathrm{thr}_{\max}+1$ and thus $|N_{G'}(v) \cap \mathcal{A}^{i-1}_{G',\mathrm{thr}}(S')| \geq \mathrm{thr}_{\max}$. Hence, $\mathrm{thr}(v) > |N_{G'}(v) \cap \mathcal{A}^{i-1}_{G',\mathrm{thr}}(S')| \geq \mathrm{thr}_{\max}$, a contradiction. ■

Since we do not know the target set S for G, two problems have to be solved in order to convert this lemma into a data reduction rule: The first problem is to find out by how much we have to decrease k, or, equivalently, how to compute $|S \cap C_1|$ in polynomial time? The second problem is that we do not know the target set S. As we show in the following, the key in overcoming these two problems is to increase the number of equivalent clusters C_j in the assumption of the lemma.

To this end, we first compute a lower and upper bound on the size of the target set for G. Let G^X be the graph that results from activating all vertices in X and applying Reduction rule 11 exhaustively. Let $C^X_1, C^X_2, \ldots, C^X_\zeta$ denote the maximal cliques of G^X. Clearly, for each clique C^X of G^X there is a cluster C in $G[V \setminus X]$ such that $C^X \subseteq C$. Let $S^X \subseteq V$ be an optimal solution for G^X. Note that S^X can be computed in linear time [90]. By construction of G^X it is clear that $|S^X|$ is a lower bound for the size of any target set for G. Furthermore, $S^X \cup X$ is a target set for G. Hence, if $k < |S^X|$ we can immediately answer no and if $k \geq |S^X| + |X| = |S^X| + \ell$ we can answer yes. Thus, we assume in the following that $|S^X| \leq k < |S^X| + \ell$. Besides these general bounds on the target set size we can also derive bounds for the number of vertices in a target set for each cluster C in $G[V \setminus X]$: If there is a (uniquely determined) clique C^X in G^X such that $C^X \subseteq C$, then set $\min(C) = |S^X \cap C^X|$. In case there is no such clique in G^X, set $\min(C) = 0$. Finally, set $\max(C) = \min\{\mathrm{thr}_{\max}, \min(C) + \ell\}$. Clearly, $\min(C)$ and $\max(C)$ are lower resp. upper bounds on the number of vertices of C that are in an optimal target set for G. Note that if two clusters C_1 and C_2 in $G[V \setminus X]$ are equivalent with respect to X, then $\min(C_1) = \min(C_2)$. Furthermore, having $\ell + 1$ clusters $C_1, \ldots, C_{\ell+1}$ in $G[V \setminus X]$ that are equivalent with respect to X, we can conclude that for any optimal target set S there is a cluster C_i, $1 \leq i \leq \ell+1$, having exactly $\min(C_1)$ vertices in the target set, since otherwise, the solution $S^X \cup X$ for G contains fewer vertices than S. Likewise, if there are $\ell + r$ clusters $C_1, \ldots, C_{\ell+r}$ that are equivalent with respect to X, then it is clear that for any optimal target set S at least r of these clusters contain exactly $\min(C_1)$ vertices of S. Hence, increasing the number of equivalent clusters to at least $\ell + \mathrm{thr}_{\max}+1$ solves the first problem.

We overcome the second problem by relaxing the condition "equivalent with respect to X and S" for the clusters $C_1, \ldots, C_{\mathrm{thr}_{\max}} \subseteq V$ to "equivalent with respect to X" and increase the number of equivalent clusters: We can assume that, out of each cluster C, at most $\max(C) \leq \mathrm{thr}_{\max}$ vertices are in a target set. Thus, there are at most $\mathrm{thr}_{\max}^{2^\ell}$ possibilities for choosing $\mathrm{thr}_{\max}$ vertices from a cluster to be in a target set: Choose at most $\mathrm{thr}_{\max}$ vertices with the highest threshold from each of the at most 2^ℓ critical cliques of the cluster. Having a set of vertices with the same closed neighborhood and the task

is to choose s of them to be in a target set, it is best to choose the s vertices with the highest thresholds [90, Observation 7]. Thus, when increasing the number of clusters that have to be equivalent with respect to X to $\ell + \text{thr}_{\max}^{2^\ell}(\text{thr}_{\max}+1)$ we can conclude with the pigeonhole principle that there are clusters $C_{i_1}, \ldots, C_{i_{\text{thr}_{\max}+1}}$ that are equivalent with respect to X and S for any target set S and each cluster C_{i_j} contains $\min(C_{i_j})$ vertices of S. Hence, applying Lemma 22 to this set we arrive at the following reduction rule.

Reduction rule 23 *Let $I = (G = (V, E), \text{thr}, k)$ be an instance of* Target Set Selection *that is reduced with respect to Reduction rule 11 and let $X \subseteq V$ be a cvd set of size ℓ. Let $C_1, C_2, \ldots, C_\alpha \subset V$ be disjoint clusters in $G[V \setminus X]$ such that $\alpha = \ell + \text{thr}_{\max}^{2^\ell}(\text{thr}_{\max}+1)$ and for each pair C_i, C_j, $1 \leq i, j \leq \alpha$, it holds that $C_i \equiv_X C_j$. Then delete C_1 and reduce k by $\min(C_1)$.*

The correctness of the data reduction rule follows from Lemma 22 and the above discussion. As to the running time, note that Reduction rule 23 can be exhaustively applied in $O(n^2)$ time. Since we require that the cvd set X is given, we can compute the clusters in $G[V \setminus X]$ in linear time. Then, we sort the vertices in these clusters by neighborhood and threshold. This can be done in $O(n \log(n))$ time. After this sorting the check whether two clusters are equivalent with respect to X can be done in linear time: Simply iterate over the sorted vertices and check whether the current vertices in both clusters have the same neighborhood and threshold. Overall, iterating over all clusters in $G[V \setminus X]$, determining the equivalent clusters, and deleting the respective clusters can be done in $O(n^2)$ time.

With these data reduction rules we now arrive at the following theorem.

Theorem 24 Target Set Selection *admits a kernel with $\text{thr}_{\max}^{O(2^\ell \text{thr}_{\max})}$ ℓ vertices, where ℓ is the cluster vertex deletion number and $\text{thr}_{\max}$ is the maximum threshold.*

Proof. Let $I = (G = (V, E), \text{thr}, k)$ be an instance of Target Set Selection that is reduced with respect to Reduction rules 11, 20, and 23. Furthermore let $X \subseteq V$ be a cvd set and let $\ell = |X|$.

Since I is reduced with respect to Reduction rule 20, the clusters in $G[V \setminus X]$ have size at most $2^\ell \text{thr}_{\max}$. Hence, there are at most $(\text{thr}_{\max}+1)^{2^\ell \text{thr}_{\max}}$ clusters in $G[V \setminus X]$ that are all pairwise **not** equivalent with respect to X. Furthermore, since I is reduced with respect to Reduction rule 23, each equivalence class of $\equiv_X$ contains at most $\ell + \text{thr}_{\max}^{2^\ell}(\text{thr}_{\max}+1)$ clusters. Thus, the number of clusters in $G[V \setminus X]$ is bounded by $(\ell + \text{thr}_{\max}^{2^\ell}(\text{thr}_{\max}+1))(\text{thr}_{\max}+1)^{2^\ell \text{thr}_{\max}}$, each of these clusters contains at most $2^\ell \text{thr}_{\max}$ vertices. Overall this gives $\text{thr}_{\max}^{O(2^\ell \text{thr}_{\max})}$ ℓ vertices in $G[V \setminus X]$ and, thus, G contains at most $\text{thr}_{\max}^{O(2^\ell \text{thr}_{\max})}$ ℓ vertices. The Reduction rules 11 and 20 can both be applied exhaustively in $O(n+m)$ time and Reduction rule 23 can be applied exhaustively in $O(n^2)$. Overall, the kernelization runs in $O(n^2)$ time. ■

Clearly this kernel implies that Target Set Selection is fixed-parameter tractable with respect to the combined parameter $\text{thr}_{\max}$ and ℓ. This yields the following result for Target Set Selection with constant thresholds.

Corollary 25 Target Set Selection *with constant thresholds is fixed-parameter tractable with respect to the parameter "cvd number".*

3.5 Complementary problem

In this section, we turn our attention to the complementary problem named Influence. First, we consider the parameterized approximability of the optimization version. As explained in the preliminaries, there are two versions of it denoted by Max Closed Influence and Max Open Influence. Notably, we prove that both versions are strongly parameterized inapproximable with respect to k even for constant and majority thresholds. However, in the unanimity case, we observe that they admit a fpt-time approximation algorithm with respect to k whereas their decision version is W[1]-hard and Max Open Influence cannot be approximated within $n^{1-\varepsilon}$ in polynomial time for all $\varepsilon > 0$ unless NP = ZPP. Second, we prove that Influence is fixed-parameter tractable with respect to the combined parameter k and maximum degree of the input graph.

3.5.1 Inapproximability results

Here, we consider the parameterized approximability of both Max Closed Influence and Max Open Influence with respect to the parameter k. We show that these problems are W[2]-hard to approximate within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ for majority thresholds and thresholds at most two, respectively.

Theorem 26 *For any $\varepsilon > 0$, Max Closed Influence and Max Open Influence with majority thresholds cannot be approximated within $n^{1-\varepsilon}$ in fpt-time with respect to the parameter k even on bipartite graphs, unless* FPT = W[2].

Proof. By Lemma 10, it suffices to show the result for Max Closed Influence. We construct a parameterized gap-introducing reduction from the W[2]-hard problem Dominating Set (see Appendix A) to Max Closed Influence with majority. In this reduction, we will make use of the *ℓ-edge* gadget, for some integer ℓ. An ℓ-edge between two vertices u and v consists of ℓ vertices of threshold one adjacent to both u and v.

Given an instance $I = (G = (V, E), k)$ of Dominating Set with $n = |V|$, $m = |E|$, we define an instance $I' = (G', k + 1)$ of Max Closed Influence with majority thresholds as follows. We start with the graph obtained from I as described in the basic reduction II (see Section 3.2.1), then we modify it to get G' as follows. Replace every edge $v^t v^b$ by an $(k+2)$-edge between v^t and v^b. Moreover, for a given constant $\beta = \frac{8}{\varepsilon} - 5$, let $L = \lceil n^\beta \rceil$ and we add nL more vertices $x_1^1, \ldots, x_n^1, \ldots, x_1^L, \ldots, x_n^L$. For $i = 1, \ldots, n$, vertex x_i^1 is adjacent to all the bottom vertices. Moreover, for any $j = 2, \ldots, L$, each x_i^j is adjacent to x_k^{j-1}, for any $i, k \in \{1, \ldots, n\}$. We also add a vertex w and an $n+(k+2)(\deg_G(v)-1)$-edge between w and v^b, for any bottom vertex v^b. For $i = 1, \ldots, n$, vertex x_i^1 is adjacent to w. For $i = 1, \ldots, n$ add n pending-vertices (*i.e.* degree one vertices) adjacent to x_i^L. For any vertex v^t add $(\deg_G(v) + 1)(k + 2)$ pending-vertices adjacent to v^t. Add also $n + n^2 + (k + 2)(2m - n)$ pending-vertices adjacent to w. All vertices of the graph G' have the majority thresholds (see also Figure 3.5). Let n' be the order of G', notice that we have $n' \leq n^4 + nL \leq n^{\beta+5}$.

We claim that if I is a yes-instance then $\mathrm{opt}(I') \geq n^{\beta+1}$; otherwise $\mathrm{opt}(I') < \frac{n^{\beta+1}}{(n')^{1-\varepsilon}}$.

Suppose that there exists a dominating set $S \subseteq V$ in G of size at most k. Consider the solution S' for I' containing the corresponding top vertices and vertex w. After

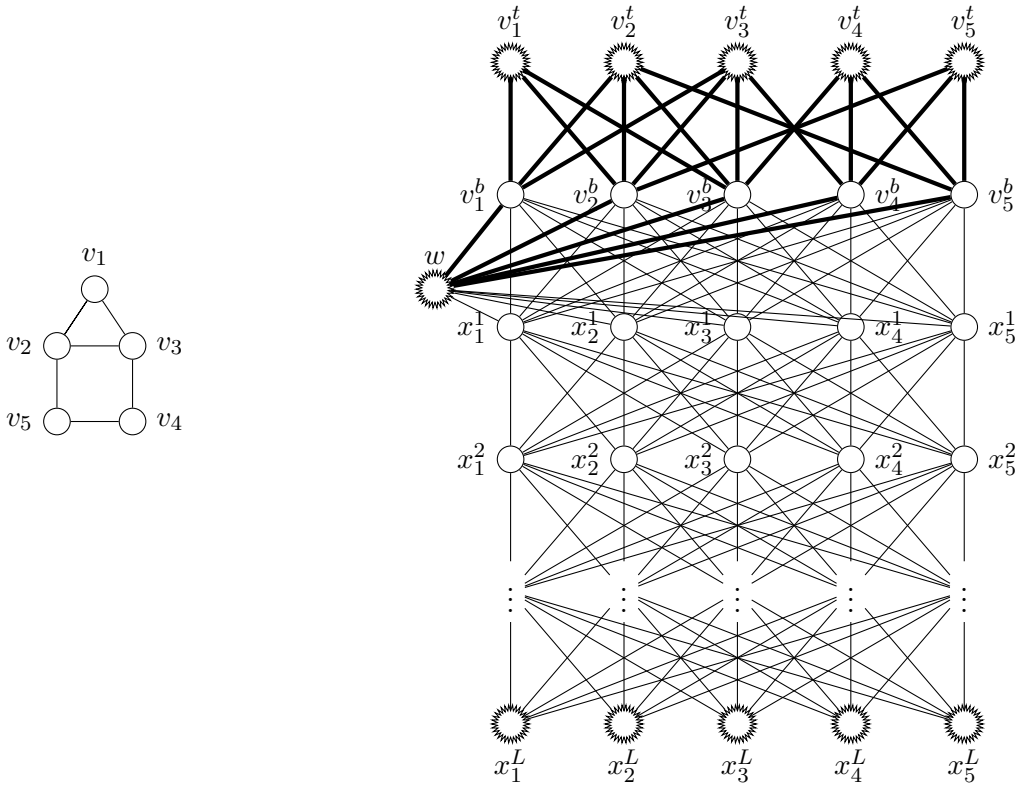


Figure 3.5: The graph G' (right) obtained from G (left) after carrying out the modifications of Theorem 26. A thick edge represents an ℓ-edge for some $\ell > 0$. A "star" vertex v represents a vertex adjacent to $\frac{\deg_{G'}(v)}{2}$ pending-vertices.

the first step, all vertices belonging to the edge gadgets which top vertex is in S' are activated. Since S is a dominating set in G, after the second step, all the bottom vertices are activated. Indeed $\deg_{G'}(v^b) = 2(n + (k+2)\deg_G(v))$ and after the first step v^b has at least $k+2$ neighbors activated belonging to an $(k+2)$-edge between v^b and some $u^t \in V$ and $n + (k+2)(\deg_G(v) - 1)$ neighbors activated belonging to an $n + (k+2)(\deg_G(v) - 1)$-edge between v^b and w. Thus, every vertex x_i^1 gets active after the third step, and generally after the jth step, $j = 4, \ldots, L+2$ the vertices x_i^{j-2} are activated, and at the $(L+3)$th step all pending-vertices adjacent to x_i^L are activated. Therefore, the size of an optimal solution is at least $nL \geq n^{\beta+1}$.

Suppose that there is no dominating set in G of size k. Without loss of generality, we may assume that no pending-vertices are in a solution of I' since they all have threshold one. If w does not take part of a solution in I', then no vertex x_i^1 could be activated and in this case $\text{opt}(I')$ is less than $n' - nL \leq n^4$. Consider now the solutions of I' of size $k+1$ that contain w. Observe that if a top-vertex v^t gets active through bottom-vertices then v^t can not activate any other bottom-vertices. Indeed, as a contradiction, suppose that v^t is adjacent to a non-activated bottom-vertex. It follows that v^t could not have been activated because of its threshold and that no pending-vertices are part of the solution, a contradiction. Notice also that it is not possible to activate a bottom vertex by selecting some x_i^1 vertices since of their threshold. Moreover, since there is no dominating set of size k, any subset of k top vertices cannot activate all bottom vertices, therefore no vertex x_i^k, $i = 1, \ldots, n, k = 1, \ldots, L$

can be activated. Hence, less than $n' - nL$ vertices can be activated in G' and the size of an optimal solution is at most $n^4 \leq \frac{n^{\beta+1}}{(n')^{1-\varepsilon}}$.

We then have a gap of $(n')^{1-\varepsilon}$, and the result follows from Theorem 9. ■

Theorem 27 *For any $\varepsilon > 0$,* Max Closed Influence *and* Max Open Influence *with thresholds at most two cannot be approximated within $n^{1-\varepsilon}$ in fpt-time with respect to the parameter k even on bipartite graphs, unless* FPT = W[2].

Proof. By Lemma 10, it suffices to prove the result for Max Closed Influence. We construct a parameterized gap-introducing reduction from the W[2]-hard problem Dominating Set (see Appendix A) to Max Closed Influence with thresholds at most two. In this reduction, we will make use of the *directed edge* gadget. A directed edge from a vertex u to another vertex v consists of a 4-cycle $\{a, b, c, d\}$ such that a and u as well as c and v are adjacent. Moreover $\mathrm{thr}(a) = \mathrm{thr}(b) = \mathrm{thr}(d) = 1$ and $\mathrm{thr}(c) = 2$. The idea is that the vertices in the directed edge gadget become active if u is activated but not if v is activated. Hence, the activation process may go from u to v via the gadget but not in the reverse direction. In the rest of the proof, we may assume that no vertices from $\{a, b, c, d\}$ are part of a solution of Max Closed Influence. Indeed, it is always as good to take the vertex u instead. We will also make use of a *directed tree* with leaves $x_1, \ldots, x_n$ and root r defined as follows: introduce $n - 1$ new vertices $y_2, \ldots, y_n$ and insert directed edges from x_1 to y_2, from x_2 to y_2, from y_i to y_{i+1}, for $i = 2, \ldots, n-1$, from x_i to y_i, for $i = 3, \ldots, n$, from y_n to r. Moreover $\mathrm{thr}(y_i) = 2$, $i = 2, \ldots, n$ and $\mathrm{thr}(r) = 1$. The idea is that the vertices in the directed tree become active if all vertices $x_1, \ldots, x_n$ are activated but not if r is activated. So, we may assume that no vertex from $y_2, \ldots, y_n$ is part of a solution of Max Closed Influence.

Given an instance $I = (G = (V, E), k)$ of Dominating Set with $n = |V|$, we define an instance $I' = (G', \mathrm{thr}, k)$ of Max Closed Influence as follows. We start with the graph and thresholds obtained from I as described in the basic reduction II (see Section 3.2.1), then we modify them to get G' and the function thr as follows. Set the thresholds of top-vertices to two. Replace every edge between a top vertex v^t and a bottom vertex v^b by a directed edge from v^t to v^b. Let $L = \lceil n^\beta \rceil$ where $\beta = \frac{4}{\varepsilon} - 3$. For $j = 1, \ldots, L$, add vertices $p_1^j, \ldots, p_n^j$ and a directed tree between leaves v_i^b, $i = 1, \ldots, n$ and root p_ℓ^1, for $\ell = 1, \ldots, n$. Moreover for $j = 1, \ldots, L-1$ add directed trees between leaves $p_1^j, \ldots, p_n^j$ and root p_ℓ^{j+1}, for $\ell = 1, \ldots, n$. This completes the construction (see Figure 3.6). Let n' be the order of G', notice that we have $n' = 2n + n^2 n^\beta + 4(2n-1)n^{\beta+1} < n^{\beta+3}$.

We claim that if I is a yes-instance then $\mathrm{opt}(I') > n^{\beta+2}$; otherwise $\mathrm{opt}(I') < \frac{n^{\beta+2}}{(n')^{1-\varepsilon}}$.

Suppose that there exists a dominating set $S \subseteq V$ in G of size at most k. Consider the solution S' for I' containing the corresponding top vertices. Since S is a dominating set in G, after the fourth step, all the bottom vertices are activated. It follows that at the end of the activation process all the vertices of the graph G' are activated except the top vertices outside S' and the vertices of some directed edges of the basic gadget. The optimum solution is $\mathrm{opt}(I') > n' - 5n^2 > n^{\beta+2}$.

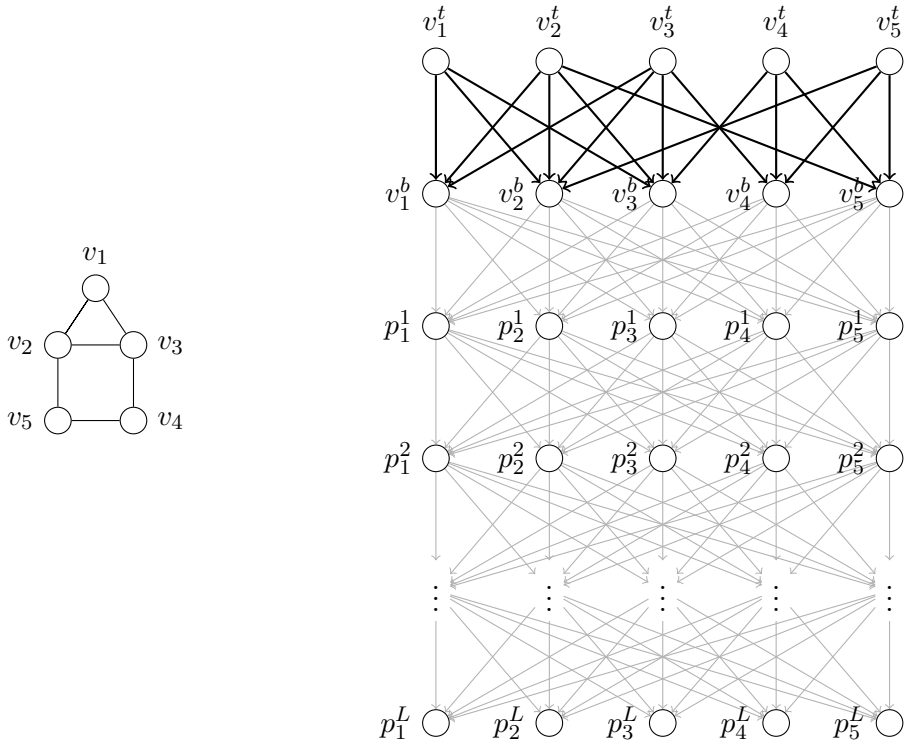


Figure 3.6: The graph G' (right) obtained from G (left) after carrying out the modifications of Theorem 27. A black arrow from u to v represents a directed edge gadget from u to v. A gray arrow from u to v indicates a directed tree where u is one of the leafs and v is the root.

Suppose that there is no dominating set in G of size k. Consider a solution S' for I' of size k. Without loss of generality, we may assume that no p_i^j vertices or bottom vertices are contained in S' since they all have threshold one. For the reason previously mentioned, we know that no vertices from the directed edge gadgets and no vertices from the directed trees are in S'. It follows that S' only contains top-vertices. Since there is no dominating set of size k in G then at least one bottom-vertex is not activated. Moreover, because of the directed edges the activated bottom-vertices cannot activate new top-vertices. Thus at least vertex of each directed tree with roots p_i^1, $i = 1, \ldots, n$ cannot be activated implying that no p_i^j vertices can be activated. This leads to a solution of size at most $5n^2 < n^3 \leq \frac{n^{\beta+2}}{(n')^{1-\varepsilon}}$.

We then have a gap of $(n')^{1-\varepsilon}$, and the result follows from Theorem 9. ■

Using Lemma 7, Theorem 26, and Theorem 27 we can deduce the following corollary.

Corollary 28 *For any strictly increasing function α,* Max Closed Influence *and* Max Open Influence *with thresholds at most two or majority thresholds cannot be approximated within $\alpha(k)$ in fpt-time with respect to the parameter k unless* FPT = W[2].

3.5.2 The unanimity case

For the unanimity thresholds case, we will give some results on general graphs before focusing on bounded degree graphs and regular graphs.

General graphs. We first show that, in the unanimity case, Influence parameterized by the combined parameter solution size k and number of newly activated vertices ℓ is W[1]-hard, and Max Open Influence is not approximable within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ in polynomial time, unless NP = ZPP. However, if we are allowed to use fpt-time then Max Open Influence with unanimity is $\alpha(n)$-approximable in fpt-time with respect to the parameter k for any strictly increasing function α.

Theorem 29 Influence *with unanimity thresholds is* W[1]*-hard with respect to the combined parameter solution size k and number of newly activated vertices ℓ even for bipartite graphs.*

Proof. We construct a fpt-reduction from the W[1]-hard problem Clique (see Appendix A) to Influence. Given an instance $(G = (V, E), k)$ of Clique, we construct an instance $(G' = (V', E'), \text{thr}, k, \ell)$ of Influence as follows. For each vertex $v \in V$ add a copy v' to V'. For each edge $uv \in E$, add $k+1$ edge-vertices $e^1_{uv}, \ldots, e^{k+1}_{uv}$ adjacent to both u' and v'. Set $\ell = (k+1)\binom{k}{2}$ and $\text{thr}(u) = \deg_{G'}(u)$ for all $u \in V'$.

We claim that there is a clique of size k in G if and only if there exists a subset $S \subseteq V'$ of size k such that $|\sigma(S)| \geq \ell$.

"⇒": Assume that there is a clique $C \subseteq V$ of size k in G. One can easily verify that the set $S = \{v' \in V' : v \in C\}$ activates $|\sigma(S)| \geq (k+1)\binom{k}{2} = \ell$ edge-vertices in G' since C is clique.

"⇐": Suppose that there exists a subset $S \subseteq V'$ of size k such that $|\sigma(S)| \geq \ell$. We may assume without loss of generality that no edge-vertices belong to S. Indeed, each edge-vertex is adjacent to only vertices with threshold at least $k+1$. Thus choosing some edge-vertices to S cannot activate any new vertices in G'. Since the solution S activates at least $(k+1)\binom{k}{2}$ edge-vertices, this implies that S is a clique in G. ■

Theorem 30 *For any $\varepsilon > 0$,* Max Open Influence *with unanimity thresholds cannot be approximated within $n^{1-\varepsilon}$ in polynomial time, unless* NP = ZPP.

Proof. We will show how to transform any approximation algorithm for Max Open Influence into another one with the same ratio for Max Independent Set (see Appendix A). Consider an instance $I = (G, k)$ of Max Open Influence with unanimity thresholds. One can note and easily check that the following holds. Given a solution $S \subseteq V$ of I, $\sigma(S)$ is obtained in only one step of the diffusion process and is an independent set. Therefore there exists an integer $k^* \in \{1, \ldots, n\}$ such that $\sigma(OPT(I_{k^*}))$ is the maximum independent set in G, where $OPT(I_{k^*})$ is the optimal solution of the instance $I_{k^*} = (G, k^*)$ of Max Open Influence (and thus $|\sigma(OPT(I_{k^*}))| = \text{opt}(I_{k^*})$).

Suppose that Max Open Influence has a polynomial-time $\alpha(n)$-approximation algorithm A, we then have $|\sigma(S_A)| \geq \frac{\text{opt}(I_{k^*})}{\alpha(n)}$, where S_A is the solution given by A when applied on the instance I_{k^*}. Furthermore, it follows from the previous observation that $\sigma(S_A)$ is an independent set in G and then an $\alpha(n)$-approximate solution of

Max Independent Set on the instance I_{k^*}.

Now, it suffices to apply the approximation algorithm A for each $k = 1, \ldots, n$ and return the approximate solution $S_{\max}$ that has the largest value. Given this solution, we have $|\sigma(S_{\max})| \geq |\sigma(S_A)|$. Hence, we get a polynomial-time $\alpha(n)$-approximation algorithm for Max Independent Set problem. Since Max Independent Set cannot be approximated within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless $\mathsf{NP} = \mathsf{ZPP}$ [66], the result follows. ■

In the following, we provide the fpt-time approximation algorithm for Max Open Influence and Max Closed Influence. As a first step toward this goal, we first give the following polynomial-time approximation algorithm.

Theorem 31 Max Open Influence *and* Max Closed Influence *with unanimity thresholds are* 2^k*-approximable in polynomial time.*

Proof. By Lemma 10, it suffices to show the result for Max Open Influence. Let $I = (G = (V, E), k)$ be an instance of Max Open Influence with unanimity thresholds. The polynomial-time algorithm consists in the following two steps: (i) Find $F \subseteq V$, the largest set of false twins (see Section 2.1) such that $\deg(v) \leq k$, $\forall v \in F$, and (ii) Return $N(F)$. The first step can be done for example by searching for the largest set of identical lines with at most k ones in the adjacency matrix of the graph. Since F is a false-twins set with vertices of degree at most k, the size of the neighborhood of F is also bounded by k. Consider the activation of the set $N(F)$. After one step, this will activate $|\sigma(N(F))| \geq |F|$ vertices, since all the neighborhood of the vertices in F are activated.

To complete the proof, observe that for any solution of size at most k, there are at most 2^k different "false-twins sets". Therefore, any optimal solution could activate at most $2^k \cdot |F|$ vertices, providing the claimed approximation ratio. ■

Using Lemma 7 and Theorem 31 we directly get the following.

Corollary 32 *For any strictly increasing function* α*,* Max Open Influence *and* Max Closed Influence *with unanimity thresholds are* $\alpha(n)$*-approximable in fpt-time with respect to the parameter* k*.*

For example, Max Open Influence is $\log(n)$-approximable in fpt-time with respect to the parameter k.

Finding dense subgraphs. In the following we show that Max Open Influence with unanimity thresholds is at least as difficult to approximate as the Densest k-Subgraph problem (see Appendix A). In particular, any positive approximation result for Max Open Influence with unanimity would directly transfers to Densest k-Subgraph.

Theorem 33 *For any strictly increasing function* r*, if* Max Open Influence *with unanimity thresholds is* $\alpha(n)$*-approximable in fpt-time with respect to the parameter* k *then* Densest k-Subgraph *is* $\alpha(n)$*-approximable in fpt-time with respect to the parameter* k*.*

Proof. We give an E-reduction from Densest k-Subgraph to Max Open Influence. Consider an instance I of Densest k-Subgraph formed by a graph $G = (V, E)$. We construct an instance I' of Max Open Influence with unanimity

thresholds consisting of graph $G' = (V', E')$ as follows. For each vertex $v \in V$ add a copy v' to V'; for each edge $uv \in E$ add a vertex e_{uv} to V', moreover add $k+1$ vertices $x_1, \ldots, x_{k+1}$. For any edge $uv \in E$ add edges $u'e_{uv}, e_{uv}v'$ to E', and add an edge between x_i and v' for any $1 \leq i \leq k+1$.

Let $S \subseteq V$, $|S| = k$ be an optimum solution for I that is $\mathrm{opt}(I)$ is the number of edges induced by S. The set $S' = \{v' : v \in S\}$ is such that $|\sigma(S')| = \mathrm{opt}(I)$ since no x vertex will be activated. Thus $\mathrm{opt}(I') \geq \mathrm{opt}(I)$.

Given any solution $S' \subseteq V'$ of size k, we can consider that S' contains only vertices of type v' such that $v \in V$. Thus the set $S = \{v : v' \in S'\}$ has value $\mathrm{cost}(I, S) = \mathrm{cost}(I', S')$. Moreover if S' is optimal, then $\mathrm{opt}(I) \geq \mathrm{opt}(I')$ and thus $\mathrm{opt}(I) = \mathrm{opt}(I')$. Therefore, we have $\varepsilon(I, S) = \varepsilon(I', S')$. ■

Using Theorem 33 and Corollary 32, we have the following corollary, independently established in [23].

Corollary 34 *For any strictly increasing function α,* Densest k-Subgraph *is $\alpha(n)$-approximable in fpt-time with respect to the parameter k.*

Bounded degree graphs. We show in the following that Max Open Influence and thus Max Closed Influence are constant approximable in polynomial time on bounded degree graphs with unanimity thresholds. Moreover, Max Closed Influence and then Max Open Influence have no polynomial-time approximation scheme even on 3-regular graphs if $\mathsf{P} \neq \mathsf{NP}$. Moreover, we show that Influence parameterized by k is in FPT.

Lemma 35 Max Open Influence *and* Max Closed Influence *with unanimity thresholds on bounded degree graphs are constant approximable in linear time.*

Proof. By Lemma 10, it suffices to show the result for Max Open Influence. Indeed on graphs of degree bounded by Δ, the optimum is bounded by $k \cdot \Delta$ and we can construct in polynomial time a solution S of value at least $\lfloor \frac{k}{\Delta} \rfloor$ by considering iteratively vertices with disjoint neighborhoods and putting their neighbors in S. ■

Theorem 36 Max Open Influence *and* Max Closed Influence *with unanimity thresholds have no polynomial-time approximation scheme even on 3-regular graphs for $k = \theta(n)$, unless $\mathsf{P} = \mathsf{NP}$.*

Proof. By Lemma 10, it suffices to show the result for Max Closed Influence. We show that if Max Closed Influence with unanimity thresholds has a polynomial-time approximation scheme $A_{\varepsilon'}$, $\varepsilon' \in (0, 1)$, on 3-regular graphs when $k = \theta(n)$, then Min Vertex Cover has also a polynomial-time approximation scheme on 3-regular graphs. Consider an instance I of Min Vertex Cover consisting of a 3-regular graph $G = (V, E)$ of order n. Clearly, a minimum vertex cover has a value $\mathrm{opt}(I)$ satisfying $\frac{n}{2} \leq \mathrm{opt}(I) < n$. For any $\varepsilon \in (0, 1)$, we apply the polynomial-time approximation scheme $A_{\varepsilon'}$ that establishes an $(1+\varepsilon')$-approximation for Max Closed Influence on graph G for each k between $\frac{n}{2}$ and n and $\varepsilon' = \frac{\varepsilon}{2-\varepsilon}$. By applying $A_{\varepsilon'}$ on G for k between $\frac{n}{2}$ and n, we obtain a solution $S_k \subset V$ of size k such that $S_k \cup \sigma(S_k)$ is an $(1+\varepsilon')$-approximation. The set $V \setminus \sigma(S_k)$ is a vertex cover in G of size denoted by val_k. We show in the following that the best solution obtained in this way is an $(1+\varepsilon)$-approximation for Min Vertex Cover on G. Indeed the best solution obtained in

this way has a value $val^* \leq val_\ell$, where val_ℓ is the value of the solution obtained for $\ell = \mathrm{opt}(I)$. Thus $val_\ell = |V \setminus \sigma(S_\ell)|$. Since $|S_\ell \cup \sigma(S_\ell)|$ is an $(1+\varepsilon')$-approximation and the optimum solution activates all vertices, we have $|S_\ell \cup \sigma(S_\ell)| \geq \frac{n}{1+\varepsilon'}$ and $|V \setminus (S_\ell \cup \sigma(S_\ell))| \leq n\frac{\varepsilon'}{1+\varepsilon'}$. Thus $val^* \leq val_\ell \leq \ell + n\frac{\varepsilon'}{1+\varepsilon'} \leq \ell(1 + \frac{2\varepsilon'}{1+\varepsilon'}) = \ell(1+\varepsilon)$. The theorem follows from the fact that Min Vertex Cover has no polynomial-time approximation scheme on 3-regular graphs, unless P = NP [6]. ■

In Theorem 29 we showed that Influence with unanimity thresholds parameterized by the combined parameter k and ℓ is W[1]-hard. In the following we give several fixed-parameter tractability results for Influence parameterized by k on regular graphs and bounded degree graphs with unanimity thresholds. First we show that using results of Cai et al. [26] we can obtain fixed-parameter tractable algorithms. Then we establish an explicit and more efficient combinatorial algorithm.

Theorem 37 Influence *with unanimity thresholds can be solved in* $2^{O(k\Delta^3)}n^2 \log n$ *time where* Δ *denotes the maximum degree and in* $2^{O(k^2 \log k)}n \log n$ *time for regular graphs.*

Proof. For graphs of maximum degree Δ, we simply apply the result from [26, Theorem 4] with $i = 3$.

Let G be a Δ-regular graph. When $\Delta > k$, any k vertices of the graph form a solution since no vertex outside the set becomes active. Hence, we assume in the following that $\Delta \leq k$. Since G is regular, it follows that any subset S, $|S| = k$ can activate at most k vertices. Hence, the graph $G[S \cup \sigma(S)]$ contains at most $2k$ vertices and, thus, $\ell \leq k$. Furthermore, since we consider unanimity thresholds, every vertex $v \in \sigma(S)$ has exactly Δ neighbors in S and, thus, $|N_{G[S\cup\sigma(S)]}(v)| = \Delta$ and $N_{G[S\cup\sigma(S)]}(v) \subseteq S$. Our fpt-algorithm solving Influence runs in two phases:
Phase 1: Guess a graph H being isomorphic to $G[S \cup \sigma(S)]$.

Phase 2: Check whether H is a subgraph of G.

Phase 1 is realized by simply iterating over all possible graphs H with $k + \ell$ vertices. A simple upper bound on the number of different graphs with $k+\ell$ vertices is $2^{\binom{k+\ell}{2}} \leq 2^{4k^2}$. Hence, in Phase 1 the algorithm tries at most $O(2^{4k^2})$ possibilities. Note that Phase 2 can be done in $2^{O(\Delta k \log k)}n \log n$ using a result from [26, Theorem 1]. Altogether this gives a running time of $O(2^{4k^2}2^{O(\Delta k \log k)}n \log n)$. Since $\Delta \leq k$, this gives $2^{O(k^2 \log k)}n \log n$. The correctness of the algorithm follows from the exhaustive search. ■

While the previous results use general frameworks to solve the problem, we now give a direct combinatorial algorithm for Influence with unanimity thresholds on bounded degree graphs. For this algorithm we need the following definition and lemma.

Definition 36: Realizing vertex

Let (α, β) be a pair of positive integers, $G = (V, E)$ a graph with unanimity thresholds. We call a vertex v a *realizing* vertex for the pair (α, β) if there exists a vertex subset $V' \subseteq N^{2\alpha-1}[v]$ of size $|V'| \leq \alpha$ such that $|\sigma(V')| \geq \beta$ and $\sigma[V']$ is connected. Furthermore, we call $\sigma[V']$ a realization of the pair (α, β).

We show first that in bounded degree graphs the problem of deciding whether a vertex is a realizing vertex for a pair of positive integers (α, β) is fixed-parameter tractable with

respect to the parameter α.

Lemma 38 *Checking whether a vertex v is a realizing vertex for a pair of positive integers (α, β) can be done in $\Delta^{O(\alpha^2)}$ time, where Δ is the maximum degree.*

Proof. The algorithm solving the problem checks for all vertex subsets V' of size α in $N^{2\alpha-1}[v]$ whether V' activates at least β vertices and whether $\sigma[V']$ is connected. Since we consider unanimity thresholds it follows that $\sigma[V'] \subseteq N^{2\alpha}[v]$.

The correctness of this algorithm results from the exhaustive search. We study in the following the running time: The $(2\alpha-1)^{\text{th}}$ neighborhood of any vertex contains at most $\Delta(\Delta^{2\alpha})/(\Delta-1)+1 \leq 2\Delta^{2\alpha}$ vertices. Hence, there are $2^{\alpha}\Delta^{(2\alpha)\alpha}$ possibilities to choose the α vertices forming V'. For each choice of V' the algorithm has to check how many vertices are activated by V'. Since this can be done in linear time and there are $O(\Delta\Delta^{2\alpha})$ edges, this gives another $O(\Delta^{2\alpha+1})$ term. Altogether, we obtain a running time of $O(2^{\alpha}\Delta^{2\alpha^2+2\alpha+1}) = \Delta^{O(\alpha^2)}$. ■

Consider in the following the Connected Influence problem that is Influence with the additional requirement that $G[\sigma[S]]$ has to be connected. Note that with Lemma 38 we can show that Connected Influence parameterized by k is fixed parameter tractable on bounded degree graphs. Indeed, observe that two vertices in $\sigma(S)$ cannot be adjacent since we consider unanimity thresholds. From this and the requirement that $G[\sigma[S]]$ is connected, it follows that $G[\sigma[S]]$ has a diameter of at most $2k$. Hence, the algorithm for Connected Influence checks for each vertex $v \in V$ whether v is a realizing vertex for the pair (k, ℓ). By Lemma 38 this gives an overall running time of $\Delta^{O(k^2)} \cdot n$.

We can extend the algorithm for the connected case to deal with the case where $G[\sigma[S]]$ is not connected. The general idea is as follows. For each connected component C_i of $G[\sigma[S]]$ the algorithm guesses the number of vertices in $S \cap C_i$ and in $\sigma(S) \cap C_i$. This gives an integer pair (k_i, ℓ_i) for each connected component in $G[\sigma[S]]$. Similar to the connected case, the algorithm will determine realizations for these pairs and the union of these realizations give S and $\sigma(S)$. Unlike the connected case, it is not enough to look for just one realization of a pair (k_i, ℓ_i) since the realizations of different pairs may be not disjoint and, thus, vertices may be counted twice as being activated. To avoid the double-counting we show that if there are "enough" different realizations for a pair (k_i, ℓ_i), then there always exist a realization being disjoint to all realizations of the other pairs. Now consider only the integer pairs that do not have "enough" different realizations. Since there are only "few" different realizations possible, the graph induced by all the vertices contained in all these realizations is "small". Thus, the algorithm can guess the realizations of the pairs having only "few" realizations and afterwards add greedily disjoint realizations of pairs having "enough" realizations. See Algorithm 1 for the pseudocode.

Theorem 39 *Algorithm 1 solves* Influence *with unanimity thresholds in $2^{O(k^2 \log(k\Delta))} \cdot n$ time, where Δ is the maximum degree of the input graph.*

Proof. Let S be a solution set, that is, $S \subset V$, $|S| \leq k$ and $\sigma(S) \geq \ell$. In the following we show that Algorithm 1 decides whether such set S exists or not in $2^{O(k^2 \log(k\Delta))} \cdot n$ time. We remark that the algorithm can be adapted to also give such set S if it exists. First we prove the correctness of the algorithm and then show the running time bound.

Correctness: We now show that a solution set S exists if and only if the algorithm returns "yes".

Algorithm 1 The pseudocode of the algorithm solving the decision problem INFLUENCE. The guessing part in the algorithm behind Lemma 38 is used in Line 7 as subroutine. The final check in Line 19 is done by brute force checking all possibilities.

```
1: procedure SOLVEINFLUENCE(G, thr, k, ℓ)
2:     Guess x ∈ {1,...,k}                          ▷ x: number of connected components of G[σ[S]]
3:     Guess (k_1,ℓ_1),...,(k_x,ℓ_x) such that Σ_{i=1}^{x} k_i = k and Σ_{i=1}^{x} ℓ_i = ℓ
4:     Initialize c_1 = c_2 = ... = c_x ← 0          ▷ one counter for each integer pair (k_i, ℓ_i)
5:     for each vertex v ∈ V do                      ▷ determine realizing vertices
6:         for i ← 1 to x do
7:             if v is a realizing vertex for the pair (k_i, ℓ_i) then   ▷ see Lemma 38
8:                 c_i ← c_i + 1
9:                 T(v, i) = "yes"
10:            else
11:                T(v, i) = "no"
12:    initialize X ← ∅                              ▷ X stores all pairs with "few" realizations
13:    for i ← 1 to x do
14:        if c_i ≤ 2 · x · Δ^{4k} then
15:            X ← X ∪ {i}
16:    for each vertex v ∈ V do                      ▷ remove vertices not realizing any pair in X
17:        if ∀i ∈ X : T(v, i) = "no" then
18:            delete v from G.
19:    if all pairs (k_i, ℓ_i), i ∈ X, can be realized in the remaining graph then
20:        return 'yes'
21:    else
22:        return 'no'
```

"⇒:" Assume that S is the solution set. Observe that $G[\sigma[S]]$ consists of at most k connected components and, thus, the guesses in Lines 2 and 3 are correct. Clearly, in the solution set S there is a realization for each pair (k_i, ℓ_i). Furthermore observe that in Line 13 it holds that $X \subseteq \{1, \ldots, x\}$ and that in the loop starting in Line 16 only vertices that cannot realize any pair corresponding to X are deleted. Hence, there exists a realization for the pairs corresponding to X in the remaining graph. Since the checking in Line 19 is done by trying all possibilities, the algorithm returns "yes".

"⇐:" Now assume that the algorithm returns "yes". Observe that this implies that in Line 19 there exists a realization for the all the pairs corresponding to X. Hence, it remains to show that for each pair (k_j, ℓ_j) where $j \in \{1, \ldots, x\} \setminus X$ there exists a realization in G. (Clearly, if all pairs are realized then the union of the realizations form the vertex set $\sigma[S]$ such that $|S| = k$.) To see that there exist realizations for these pairs observe the following: The $(4k)^{\text{th}}$ neighborhood of any vertex contains at most $2\Delta^{4k}$ vertices. Thus, if in the case of two pairs $(k_1, \ell_1), (k_2, \ell_2)$ the value of the second counter is $c_2 > 2\Delta^{4k}$, then we can deduce that for every realizing vertex v_1 for (k_1, ℓ_1) there exists a realizing vertex v_2 for (k_2, ℓ_2) such that the distance d between v_1 and v_2 is more than $4k$. Since $d > 4k$, it follows that the realizations for (k_1, ℓ_1) and (k_2, ℓ_2) do not overlap. (If two realizations would overlap then some vertices in $\sigma(S)$ may be counted twice.) Generalizing this argument to x integer pairs $(k_1, \ell_1), \ldots, (k_x, \ell_x)$ yields the following: If there exists an $i \in \{1, \ldots, x\}$ such that $c_i > x \cdot 2 \cdot \Delta^{4k}$, then for any realization of the pairs (k_j, ℓ_j) with $i \neq j$ there exists a non-overlapping realization of (k_i, ℓ_i). Thus, we can ignore the pair (k_i, ℓ_i) where $c_i > x \cdot 2 \cdot \Delta^{4k}$ in the remaining algorithm and can assume that (k_i, ℓ_i) is realized.

Observe that from the Lines 5 to 16 it follows that for all $j \in \{1, \ldots, x\} \setminus X$ we have $c_j > x \cdot 2 \cdot \Delta^{4k}$. Thus, from the argumentation in the previous paragraph

it follows that there exist non-overlapping realizations for all pairs corresponding to $\{1, \ldots, x\} \setminus X$. Thus, there exists a solution set S as required.

Running time: Observe that $\ell \leq \Delta k$ as described in the proof of Lemma 35. Thus, the guessing in Lines 2 and 3 can clearly be done in $O(k \cdot k^k(\Delta k)^k) = O(k^{2k+1}\Delta^k)$. By Lemma 38 the checking in Line 7 can be done in $\Delta^{O(k_i^2)}$ time. Thus, the loop in Line 5 requires $n \cdot \sum_{i=1}^{x} \Delta^{O(k_i^2)} \leq \Delta^{O(k^2)} \cdot x \cdot n$ time. Clearly, the loop in Line 13 needs $O(x) \leq O(k)$ time. Furthermore, the loop in Line 16 needs $O(k \cdot n)$ time. For the checking in Line 19 observe the following. After deleting the vertices in the loop in Line 16 the remaining graph can have at most $\sum_{i \in X} c_i \leq x \cdot 2 \cdot x \cdot \Delta^{4k}$ vertices. Furthermore, $\sum_{i \in X} k_i \leq k$ and, thus, there are at most $(2 \cdot x^2 \cdot \Delta^{4k})^k$ candidate subsets for the solution set S. Checking whether $\sum_{i \in X} k_i$ chosen vertices activate $\sum_{i \in X} \ell_i$ other vertices can be done in $(2 \cdot x^2 \cdot \Delta^{4k})^2$ time. Hence, the checking in Line 19 can be done in $\Delta^{O(k^2)}$ time. Putting all together we arrive at a running time of $(k\Delta)^{O(k^2)} \cdot n = 2^{O(k^2 \log(k\Delta))} \cdot n$. ■

3.6 Conclusion and open problems

In this chapter, we investigate the parameterized complexity of Target Set Selection with respect to several graph parameters. We also established results concerning the polynomial-time and fpt-time approximability of two related problems, namely Max Open Influence and Max Closed Influence. We conclude this chapter with the following research directions.

(1) The parameterized tractability of Target Set Selection with respect to "cluster vertex deletion number" for majority thresholds, "distance to clique" for general thresholds, and "clique cover number" for majority and constant thresholds are open.

(2) Determine whether, in terms of computational complexity, Target Set Selection with majority thresholds is basically as hard as for general thresholds but significantly easier for constant thresholds. This last fact does not hold from the polynomial-time approximation and parameterized approximation point of view.

(3) It would be interesting to incorporate further natural parameters as the "graph diameter" or the "number of activation steps".

(4) An interesting open question consists of determining whether Max Open Influence is constant approximable in fpt-time. Such a positive result would improve the approximation in fpt-time for Densest k-Subgraph.

(5) Another interesting open question is to study the approximation of Min Target Set Selection (the minimization version of Target Set Selection) in fpt-time with respect to the optimum value or other structural parameters.

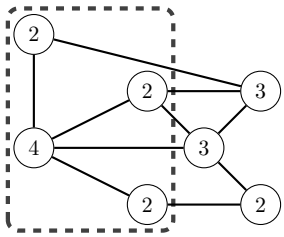
(6) Due to applications in social networks, the identification of tractable special cases in scale-free graphs, that is, graphs with power law degree distributions, would be of particular interest.

Chapter

4 Finding harmless individuals

In the previous chapter, we tried to find a subset of the most "influential" vertices. We now ask the natural converse question: find the maximum subset of "less influential" vertices, namely a *harmless set*. Informally speaking, the problem asks to find the largest subset of vertices with the property that if any set of vertices gets activated in it then there will be no propogation at all *i.e.* no new vertex can be activated by the application of the same propagation rule as for Target Set Selection (see Chapter 3). In this context, the threshold of a vertex represents its reliability regarding its neighborhood; that is, how many neighbors need to be activated before a vertex gets himself actived.

In the graph below, if any vertex gets activated inside the dashed rectangle (harmless set) then no new vertices are activated. The thresholds are indicated inside the vertices.



In this chapter, we study the parameterized complexity and the approximability of the problem.

The content of this chapter is based on in the following paper.

▶ C. Bazgan and M. Chopin, *The robust set problem: parameterized complexity and approximation*, Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS 2012), LNCS 7464, pp. 136–147, 2012.

4.1 Introduction

Previously, we emphasize the strong intractability nature of maximizing the spread of information through networks. Therefore, with regards to this hardness, it is an interesting question to ask the complexity of the converse problem: find the largest subset of vertices such that if any set of vertices gets activated in it then there will be no consequence for all the other vertices. In this context, there is no propagation process involved like previously stated. On the contrary we want to prevent such phenomenon here. More formally, we introduce the Harmless Set problem defined as follows. Given the same input and propagation rule as for Target Set Selection (see Chapter 3), the task is to determine whether there exists a subset of vertices, called *harmless* set, of size at least k

such that every vertex v of the input graph has less than $\text{thr}(v)$ neighbors in this set. The reason for this definition follows from the fact that if any subset of a harmless set is activated then there will be no propagation at all since every vertex has a number of activated neighbors below its threshold. Another motivation for this problem arises from the context of containing the spread of dangerous ideas or epidemics which is also a fruitful research area [74, 54, 31] and further justifies the meaning of harmless set. Finally, it is worth pointing out that our problem can also be related to the generalized domination problem (σ, ρ)-Dominating Set introduced by Telle [100]: Given a graph $G = (V, E)$, two sets of non-negative integers σ and ρ, and an integer $k \geq 1$, the objective is to find a set $S \subseteq V$ of size at most k such that $|S \cap N(v)| \in \sigma$ for every vertex $v \in S$, and $|S \cap N(v)| \in \rho$ for every vertex $v \notin S$. The set S is then called a (σ, ρ)-dominating set of size at most k. As a matter of fact, if all thresholds are equal then Harmless Set is equivalent to (σ, ρ)-Dominating Set Of Size k [61] (this version asks to find a (σ, ρ)-dominating set of size exactly k) where $\sigma = \rho = \{0, \ldots, \text{thr}_{\max}\}$ and $\text{thr}_{\max}$ is the maximum threshold value. Since this last problem is in W[1] [61], this implies that Harmless Set is in W[1] if all thresholds are equal. In this work, we extend this result to the case where the thresholds are bounded by a constant. Up to our knowledge, this is the only known result that could possibly carry over.

We study the parameterized complexity of Harmless Set and the approximation of the associated maximization problem (denoted Max Harmless Set). The chapter is organized as follows. In Section 4.2 we give the definitions and terminology. In Section 4.3 we establish parameterized complexity results with respect to various threshold functions (see Table 4.1). We show that the parametric dual problem (denoted Dual Harmless Set and asks the existence of a harmless set of size at least $n - k$ where n is the size of the graph) is fixed-parameter tractable for a large family of threshold functions. In Section 4.4 we give polynomial-time algorithms to solve the problem for graphs of bounded treewidth. In Section 4.5, we turn our attention to the maximization version. We establish that it is hard to approximate the problem within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ even when all thresholds are at most two. If each threshold is equal to the degree of the vertex, we show that the problem is APX-complete. Moreover it has a polynomial-time approximation scheme on planar graphs. Conclusion and open problems are given in Section 4.6.

	Harmless Set	Dual Harmless Set
General thresholds	W[2]-complete (Th.42)	W[2]-hard
Constant thresholds	W[1]-complete (Th.44)	FPT (Th.48)
Majority thresholds	W[1]-hard (Th.44)	FPT (Th.48)
Unanimity thresholds	FPT (Th.46)	W[2]-hard*

Table 4.1: Our parameterized complexity results for Harmless Set and its parametric dual Dual Harmless Set where the parameter is k for both problems. The result marked with * is due to the equivalence between Dual Harmless Set and the Total Dominating Set problem proved W[2]-hard for parameter k in [61].

4.2 Problem definitions & terminology

Before defining the problems, we have to specify the notion of harmless vertices in a graph.

Definition 37: Harmless vertices

Let $G = (V, E)$ be a graph, and $\text{thr} : V \to \mathbb{N}$ a threshold function.
A subset $V' \subseteq V$ is called *harmless* if for every vertex $v \in V$,

$$|N(v) \cap V'| < \text{thr}(v)$$

We define in the following the problems we study in this chapter.

HARMLESS SET
Input: A graph $G = (V, E)$, a threshold function $\text{thr} : V \to \mathbb{N}$ where $1 \leq \text{thr}(v) \leq \deg(v)$ for every $v \in V$, and an integer k.
Question: Is there a harmless set $V' \subseteq V$ of size at least k?

We also consider the parametric dual problem DUAL HARMLESS SET which asks for the existence of a harmless set of size at least $n-k$ where n denotes the number of vertices in the input graph.

The optimization version of HARMLESS SET is defined as follows.

MAX HARMLESS SET
Input: A graph $G = (V, E)$ and a threshold function $\text{thr} : V \to \mathbb{N}$ where $1 \leq \text{thr}(v) \leq \deg(v)$ for every $v \in V$.
Output: A harmless set $V' \subseteq V$ such that $|V'|$ is maximized.

As in the previous chapter, we consider majority thresholds *i.e.* $\text{thr}(v) = \lceil \frac{\deg(v)}{2} \rceil$ for each vertex v, unanimity thresholds *i.e.* $\text{thr}(v) = \deg(v)$ for each vertex v, and constant thresholds *i.e.* $\text{thr}(v) \leq c$ for each vertex v and some constant $c > 1$.

4.3 Parameterized complexity

In this section, we consider the parameterized complexity of HARMLESS SET. In some reductions we make use of the following gadget: a *forbidden edge* denotes an edge uv where both vertices have threshold one. Attaching a forbidden edge to a vertex w means to create a forbidden edge uv and make w adjacent to u. Notice that none of the three vertices u, v or w can be part of a harmless set. Moreover, we need the following simple but useful reduction rule.

Reduction rule 40 *Let (G, thr, k) be an instance of* HARMLESS SET. *If there is a vertex v such that $\text{thr}(v) > k+1$ then set the threshold $\text{thr}(v)$ to $k+1$ to get a new equivalent instance (G, thr', k).*

To see that the above rule is correct, observe that if $S \subseteq V$ is a harmless set of size at least k for (G, thr, k), then any subset of size k of S is a harmless set for (G, thr', k). The converse is clear.

We now show that HARMLESS SET parameterized by k belongs to W[2] using the *Turing way*, that is, we reduce HARMLESS SET to the SHORT MULTI-TAPE NONDETERMINISTIC TURING MACHINE problem (see Section 2.3.3).

Theorem 41 HARMLESS SET *is in* W[2] *with respect to the parameter k.*

Proof. We construct an fpt-reduction from HARMLESS SET to SHORT MULTI-TAPE NONDETERMINISTIC TURING MACHINE as follows. Let (G, thr, k) be an instance of HARMLESS SET with $G = (V, E)$ and $V = \{v_1, \ldots, v_n\}$. First, exhaustively apply

Reduction rule 40 to obtain the new equivalent instance (G, thr', k). Then construct the following Turing machine M from (G, thr', k) (see Figure 4.1). We create $n+1$ tapes denoted by $T_0, T_{v_1} \ldots, T_{v_n}$. The tapes alphabet is $V \cup \{\times\}$ plus the blank symbol $\square$. Initially, every tape is filled with $\square$. The transition function is defined hereafter. First, M non-deterministically chooses k vertices and write them on tape T_0, that is, if M picks up a vertex $v \in V$ then it writes symbol v on T_0 and move T_0's head one step to the right. The previous procedure is done in $k+1$ steps. Next, for each $i = 1, \ldots, k+1$, the Turing machine writes a symbol $\times$ on each tape T_j and moves T_j's head one step to the right if v_j has a threshold greater or equal to i. According to Reduction rule 40, this phase takes at most $k+1$ steps. During the third phase, M checks whether the selected set is a harmless set as follows. First, the machine moves all heads one step to the left. If T_0's head reads symbol v then for every $u_i \in N(v)$, we simply move T_i's head one step to the left. We repeat the previous procedure until T_0's head reads a blank symbol. If all the other tapes read a $\times$ symbol then M goes in accepting state; otherwise it goes to a rejecting state. This checking phase can be done in at most $k+1$ steps. Finally, the input word x is empty and $k' = 3k+3$. It is not hard to see that (G, thr', k) is a yes-instance if and only if M accepts in at most k' steps. ■

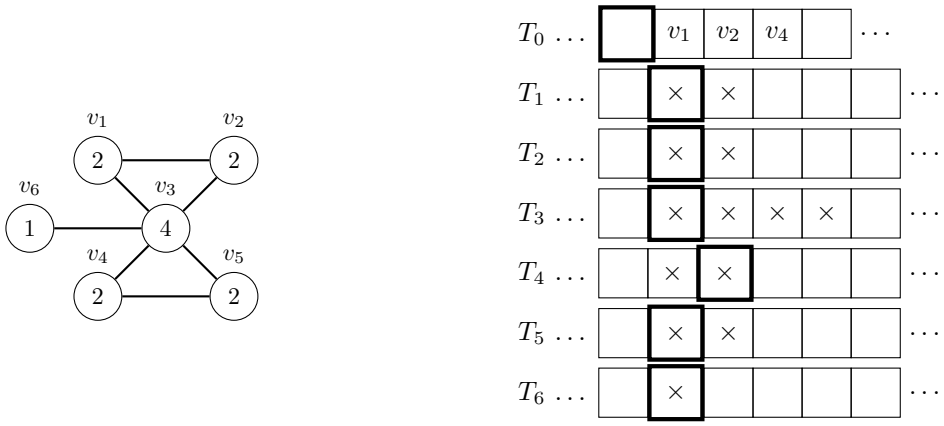


Figure 4.1: A non-deterministic multi-tape Turing machine (right) accepting an instance of Harmless Set where $k = 3$ (left). The guessed solution is the set $\{v_1, v_2, v_4\}$. We indicate the thresholds inside the vertices.

Now in order to prove the W[2]-hardness of Harmless Set paramterized by k, we construct a simple fpt-reduction from the W[2]-hard problem Red/Blue Dominating Set (see Appendix A).

Theorem 42 Harmless Set *is* W[2]*-complete with respect to the parameter* k *even on bipartite graphs.*

Proof. Membership follows from Theorem 41. Now, let us show the W[2]-hardness. Given (G, k) an instance of Red/Blue Dominating Set, we construct an instance $(G' = (V', E'), \text{thr}, k)$ of Harmless Set as follows. We consider the graph $\bar{G}$ obtained from G as follows. Two vertices $u \in R$ and $v \in B$ are adjacent in $\bar{G}$ if and only if they are not adjacent in G. Moreover, the sets R and B remain independent sets. The graph G' is obtained from this last graph by attaching $\max\{k - \deg_{\bar{G}}(v), 1\}$ forbidden edges to each vertex $v \in B$. Finally, set $\text{thr}(v) = k$ for every vertex $v \in B$

and $\text{thr}(v) = 1$ for every vertex $v \in R$. Adding several forbidden edges to the vertices of B make sure that the threshold of these vertices is less than or equal to their degree as required in the problem definition.

We claim that (G, k) is a yes-instance if and only if $(G' = (V', E'), \text{thr}, k)$ is a yes-instance.

"$\Rightarrow$": Assume that (G, k) has a solution $R' \subseteq R$ of size k. One can see that R' is also a solution for (G', thr, k) since every vertex in B is not adjacent to at least one vertex in R'.

"$\Leftarrow$": Conversely, suppose that there is a harmless set $S \subseteq V'$ of size k in G'. Since S is harmless, S cannot contain any vertex from B because of the forbidden edges, and thus S is entirely contained in R. Moreover, every vertex v in B is adjacent in G' to at most $\text{thr}(v) - 1 = k - 1$ vertices in S. Hence, every vertex in B is adjacent in G to at least one vertex in S. Therefore, S is a solution of size k for (G, k). ■

In the next two theorems, we show that Harmless Set parameterized by k goes one level down in the W-hierarchy when all thresholds are bounded by a constant.

Theorem 43 Harmless Set *with constant thresholds is in* W[1] *with respect to the parameter* k.

Proof. Let (G, thr, k) be an instance of Harmless Set where $\text{thr}(v) \leq c$, $\forall v \in V$ for some constant $c > 0$. We construct in $O(n^c)$ time, where n is the number of vertices of G, a boolean circuit $\mathcal{C}$ (see Definition 18) of depth 3 and weft 1 as follows. We identify the inputs of the circuit with the vertices of G. Connect a gate NOT to every input. For all $v \in V$ and all subsets $S' \subseteq N(v)$ of size $\text{thr}(v)$, add a gate OR connected to every gate NOT adjacent to an input in S'. Finally, add a large gate AND connected to every gate OR. It is not hard to see that G admits a harmless set of size k if and only if there is an assignment of weight k that satisfies $\mathcal{C}$. ■

We establish the W[1]-hardness of Harmless Set parameterized by k by an fpt-reduction from the W[1]-hard problem Clique (see Appendix A).

Theorem 44 Harmless Set *is* W[1]*-complete with respect to the parameter* k *even*

1. *For bipartite graphs and constant majority threshold.*
2. *For split graphs and threshold two for every vertex.*

Proof. Membership follows from Theorem 43. We now prove the W[1]-hardness.

(1): Given $(G = (V, E), k)$ an instance of Clique, we construct an instance $(G' = (V', E'), \text{thr}, k)$ of Harmless Set as follows. V' is obtained from V by adding for each edge $uv \notin E$, an edge-vertex e_{uv} and edges ue_{uv} and $e_{uv}v$ to E'. Remove every edge in E. Finally, attach a forbidden edge $p_{uv}q_{uv}$ to each edge-vertex e_{uv}. Set $\text{thr}(v) = \lceil \frac{\deg(v)}{2} \rceil, \forall v \in V'$.

We claim that (G, k) is a yes-instance if and only if (G', thr, k) is a yes-instance.

"$\Rightarrow$": Let $C \subseteq V$ be a clique of size at least k. Then C is clearly a harmless set in G' since no edge-vertex has more than one neighbor in C.

"$\Leftarrow$": Conversely, let $C' \subseteq V'$ be a harmless set in G'. Because of the forbidden edges, C' cannot contain an edge-vertex e_{uv} and p_{uv}, q_{uv} and thus $C' \subseteq V$. Moreover, since $\text{thr}(e_{uv}) = 2$, C' cannot contain u and v such that $uv \notin E$ and thus C' is a clique of size at least k in G.

(2): Let $(G = (V, E), k)$ be an instance of Clique, we construct an instance $(G' = (V', E'), \mathrm{thr}, k)$ of Harmless Set as follows. As previously, for each edge $uv \notin E$, add an edge-vertex e_{uv} and the edges ue_{uv} and $e_{uv}v$. Add edges to make the set of all edge-vertices a clique. Remove every edge in E. Finally, set $\mathrm{thr}(v) = 2$ for all $v \in V'$. Without loss of generality we may assume that $k \geq 2$ and every vertex in V has minimum degree two.

We claim that (G, k) is a yes-instance if and only if (G', thr, k) is a yes-instance.

"$\Rightarrow$": Let $C \subseteq V$ be a clique of size at least k. One can easily verify that C is a harmless set in G'.

"$\Leftarrow$": Conversely, suppose that there is a harmless set $C' \subseteq V'$ of size k. Notice that $C' \subseteq V$ since otherwise we would not have been able to take more than one vertex in G'. Indeed, if there are two vertices $u, v \in C'$ with $v \in V' \setminus V$ then there is always a vertex $w \in V' \setminus V - \{u, v\}$ adjacent to both v and u. Thus, C' is entirely contained in V. From now on, it is not hard to see that C' is a clique of size at least k in G. ■

It is interesting to note that the ratio between the number of vertices with unbounded threshold over the number of vertices with bounded threshold in the proof of Theorem 42 can be made arbitrarily small by adding many forbidden edges. This implies a sharp dichotomy between the W[2]- and W[1]-completeness of Harmless Set parameterized by k regarding the thresholds.

Unanimity threshold. We consider now the Harmless Set problem with unanimity thresholds. First, we start with the following easy observation. In the case of unanimous threshold, any harmless set is the complement of a total dominating set. Recall that a total dominating set S is a set of vertices such that every vertex in the input graph has at least one neighbor in S. Moreover, we have the following theorem.

Theorem 45 (Cockayne et al. [43]) *If G is a connected graph of order at least 3 then there is a total dominating set of size at most $2n/3$.*

This implies that, in the unanimity case, there is always a harmless set of size at least $n/3$ when $n \geq 3$. The consequence of this result is that we directly get a linear kernel of size $3k$.

Theorem 46 Harmless Set *with unanimity thresholds admits a kernel with $3k$ vertices.*

Indeed, let (G, k) be an instance of Harmless Set with unanimity thresholds, if $k \leq n/3$ then the answer is yes. If $k > n/3$ then the instance (G, k) is a kernel of size at most $3k$. However, the parameter k is "large" in this last case. This suggests to look for other parameterizations. One possibility is to decide the existence of solutions of size at least $\lceil \frac{n}{3} \rceil + k$. Another one is to decide the existence of solutions of size at least $n - k$. We study in the following this last problem.

Parametric dual. We show that Dual Harmless Set parameterized by k is in FPT for a large family of threshold functions. Toward this goal, we provide a kernelization by making use of the following reduction rule.

Reduction rule 47 *Let (G, thr, k) be an instance of* Dual Harmless Set. *If there is a vertex v such that $\deg(v) \geq k + \mathrm{thr}(v) - 1$ then remove v and decrease by one the threshold of every vertex in $N(v)$ to get a new equivalent instance (G', thr', k).*

Regarding the correctness of the above rule, let S be a harmless set of size at least $n-k$. Notice that if there is a vertex v with $\deg(v) \geq k + \text{thr}(v) - 1$ then v must be in S since otherwise it will have at most $k-1$ neighbors outside S and then at least $\text{thr}(v)$ neighbors in S.

We can now state the main result.

Theorem 48 Dual Harmless Set *admits a kernel with $O(k^2)$ vertices if for every vertex v in the input graph $\text{thr}(v) = \lceil \alpha_v \deg(v)^{\beta_v} + \gamma_v \rceil$ for any constants $\alpha_v, \beta_v \in [0,1], \alpha_v\beta_v \neq 1$, and $\gamma_v \in Q$.*

Proof. Let (G, thr, k) be an instance of Dual Harmless Set. Exhaustively apply Reduction rule 47 to get (G', thr', k). Assume that there exists a solution $S \subseteq V$ of size at least $n-k$. Because of Reduction rule 47, we have

$$\deg(v) < k + \text{thr}(v) - 1 = k + \lceil \alpha_v \deg(v)^{\beta_v} + \gamma_v \rceil - 1 \leq k + \alpha_v \deg(v)^{\beta_v} + \gamma_v \quad (4.1)$$

We claim that $\deg(v) \leq \theta_v(k)$ for all $v \in V'$ where

$$\theta_v(k) = \begin{cases} k + \alpha_v + \gamma_v & \text{if } \beta_v = 0 \\ \frac{k+\gamma_v}{1-\alpha_v} & \text{if } \beta_v = 1 \\ \frac{k+\gamma_v}{1-\beta_v} + (1/\beta_v)^{\frac{1}{1-\beta_v}} & \text{otherwise} \end{cases}$$

The first two cases are straightforward. Suppose now that $\beta_v \in (0,1)$. First, it is not hard to show that the following holds: $x^\varepsilon \leq \varepsilon x$ if and only if $x \geq (1/\varepsilon)^{\frac{1}{1-\varepsilon}}$ for any $x \geq 1$ and $\varepsilon \in (0,1)$. Hence, if $\deg(v) \geq (1/\beta_v)^{\frac{1}{1-\beta_v}}$ then, together with (Eq. 4.1), we obtain $\deg(v) \leq k + \alpha_v\beta_v \deg(v) + \gamma_v$ and thus $\deg(v) \leq \frac{k+\gamma_v}{1-\alpha_v\beta_v} \leq \theta_v(k)$. Otherwise $\deg(v) < (1/\beta_v)^{\frac{1}{1-\beta_v}} \leq \theta_v(k)$.
Since every vertex from S has at least one neighbor in $V' - S$ then $|S|$ has at most $|V' - S| \deg_{\max} \leq k\theta_{\max}(k)$ vertices where $\theta_{\max}(k) = \max_{v \in V'} \theta_v(k)$ and $\deg_{\max}$ is the maximum degree of vertices in $V' - S$.

The kernelization procedure is then defined as follows. From an instance (G, thr, k) of Dual Harmless Set, exhaustively apply Reduction rule 47 to get an instance (G', thr', k). If $|V'| > k\theta_{\max}(k) + k$ then return a trivial no-instance. Otherwise, return the instance (G', thr', k). ■

Notice that if $\alpha_v = \beta_v = 1$ and $\gamma_v = 0, \forall v \in V$ then the Dual Harmless Set problem is exactly the Total Dominating Set problem which is known to be W[2]-hard [61].

4.4 Algorithms for trees and tree-like graphs

In this section we establish a $O(\text{thr}_{\max}^{3\,\text{tw}}\, n)$-time algorithm for Max Harmless Set and a $O((k+1)^{3\,\text{tw}} n)$-time algorithm for Harmless Set where $\text{thr}_{\max}$ is the maximum threshold and tw the treewidth of the input graph. We decided to describe first a linear-time algorithm for trees. Besides to be more efficient in this case, it will introduce the underlying ideas used later for the general algorithm.

Theorem 49 Max Harmless Set *is solvable in linear time on trees.*

Proof. Let $(T = (V, E), \text{thr})$ be an instance of Max Harmless Set where T is a tree rooted at r. We describe a dynamic programming algorithm as follows. We

denote by T_v the subtree of T rooted at v. Moreover, we denote by $C(v)$ the set of children of a vertex v in T. For each $v \in V$ and each $b \in \{0,1\}$, we compute $A_v[b]$, the optimal solution for the subtree T_v with the additional constraint that the threshold of v is set to $\text{thr}(v) - b$. Set $A_v[0] = \{v\}$ and $A_v[1] = \emptyset$ for every leaf v of T. Then for all $v \in V$ and all $b \in \{0,1\}$, we compute $A_v[b]$ as follows

$$A_v[b] = \underset{S \in \{I_v[b], E_v[b]\}}{\arg\max} |S| \tag{4.2}$$

where $I_v[b]$ (resp. $E_v[b]$) is the optimum solution of T_v when $\text{thr}(v)$ is decreased by b and v is included (resp. excluded). They are computed as follows

$$I_v[b] = \begin{cases} \emptyset & \text{if } \exists c \in C(v) : A_c[0] = \emptyset \\ \{v\} \cup \bigcup_{c \in C(v) \setminus A_c[1]} A_c[1] \cup \bigcup_{i=1}^{\text{thr}(v)-b-1} B_{\pi_v(i)}[1] & \text{otherwise} \end{cases} \tag{4.3}$$

$$E_v[b] = \bigcup_{c \in C(v) \setminus A_c[0]} A_c[0] \cup \bigcup_{i=1}^{\text{thr}(v)-b-1} B_{\pi_v(i)}[0] \tag{4.4}$$

In the above formulas, for any $v \in V$ and $b \in \{0,1\}$, we set $B_v[b] = A_v[b]$ if $v \in A_v[b]$, $\emptyset$ otherwise. Furthermore, we denote by $\pi_v : \{1, \ldots, |C(v)|\} \to C(v)$ the permutation of the children of v such that $|B_{\pi_v(1)}[b]| \geq \ldots \geq |B_{\pi_v(|C(v)|)}[b]|$ for $b \in \{0,1\}$. To get the optimal solution for the tree T, return $A_r[0]$.

To see that the above algorithm is correct, first observe that when we make a decision for a vertex v, we know the decision made for all its children but we do not know about its parent, denoted by p_v. We then have to compute two optimal solutions for the subtree T_v by considering two cases: one where p_v is in the solution and the other one when it is not. The first case can be done by computing an optimal solution with the threshold of v set to $\text{thr}(v) - 1$ *i.e.*, $b = 1$. For the second case, we compute another optimal solution without modifying v's threshold *i.e.*, $b = 0$. Notice that in each case, the optimal solution for T_v takes either v (Eq. 3) or not (Eq. 4). We then keep the best of these two solutions (Eq. 2). We now explain Eq. 3 (the other equation works analogously). First, observe that if there is a child c of v such that $A_c[0] = \emptyset$ then this means that T_c has only vertices with thresholds one and thus there exists no harmless set in T if v is part of the solution. Thus we set $I_v[b] = \emptyset$. If it is not the case then the optimal solution for T_v is the union of the following three sets: the singleton $\{v\}$ since v is forced to be in the solution; the optimal solution of each subtree rooted at a child of v with the condition that this child is not taken; and the $(\text{thr}(v) - b - 1)$-th largest solutions of the subtrees rooted at a the children of v that belong to the solution. ■

Now, we present the general algorithm for solving Max Harmless Set.

Theorem 50 Max Harmless Set *is solvable in* $O(\text{thr}_{\max}^{3\,\text{tw}} \cdot n)$ *time where* $\text{thr}_{\max}$ *is the maximum threshold and* tw *is the treewidth of the input graph.*

Proof. Let $(G = (V, E), \text{thr})$ be an instance of Max Harmless Set. Assume that we are given a nice tree decomposition $\mathcal{T} = (T = (X, F), \mathcal{H})$ of G of width at most tw that can be found in fpt-time with respect to the treewidth [22]. Let T_x be the subtree

of T rooted at some node $x \in X$. We denote by $G_x = (V_x, E_x)$ the subgraph induced by the vertices from $\bigcup_{y \in T_x} H_y$. We describe a dynamic programming algorithm to solve (G, t, k) using $\mathcal{T}$.

Description. The general idea of the algorithm is as follows. For each node $x \in X$, we store a set of optimal solutions for the subgraph G_x in a table denoted by A_x. These tables are updated using a bottom-up procedure that starts from the leaves and ends to the root of T. In order to update the table of a node, we need to define an "update rule" according to the type of the node, i.e., *insert*, *forget* and *join*. Moreover, we also have to know what kind of solutions we should store in each table, or, equivalently, what information do we need in a node to update the table of its parent. In our algorithm, we use a two-entries table $A_x[\vec{t}, \vec{c}]$ where $\vec{t} : H_x \to \{1, \ldots, \text{thr}_{\max}\}$ and $\vec{c} : H_x \to \{0, 1\}$. An entry $A_x[\vec{t}, \vec{c}]$ corresponds to a maximum harmless set in G_x that contains all the vertices in $\{v \in H_x : \vec{c}(v) = 1\}$ and by imposing the threshold $\text{thr}(v) = \vec{t}(v)$, $\forall v \in H_x$. We set $A_x[\vec{t}, \vec{c}] = \emptyset$, if no such harmless set is possible. To see why this table is sufficient for our purpose, consider the updating step occuring in a join node. Let $x \in X$ be a join node with children y and z such that $H_x = H_y = H_z$. The nodes y and z have their respective tables A_y and A_z already computed by dynamic programming and we want to compute the table A_x. First, we introduce the role of the function $\vec{t}$. Let $B = N_G(H_x) \setminus V_x$. Notice that, at the current stage of the algorithm, we do not know what vertices in B will be in the maximum harmless set. Thus, to compute the table A_x one has to take into consideration that any subset $S \subseteq B$ might be in the optimal solution. Hence, we have to compute a maximum harmless set in G_x for each subset $S \subseteq B$ considering S as part of a harmless set. This can be seen as equivalent to finding a maximum harmless set in G_x for every possible thresholds $\vec{t} \in \{1, \ldots, \text{thr}_{\max}\}^{H_x}$ of vertices in H_x. Indeed, picking a vertex v from G into a harmless set is equivalent of removing v from G and decreasing the thresholds of the vertices in $N(v)$ by one. Moreover every set S only affects vertices in H_x since no vertex in B can be adjacent to a vertex in $V_x \setminus H_x$. Hence we can compute A_x solely from tables A_y and A_z. To do this, we have to take care of the following fact. Consider the optimal solutions S_y and S_z in G_y and G_z respectively and by imposing some thresholds $\vec{t}$ to H_y and H_z. Notice that we cannot directly make the union of S_y and S_z to compute the optimal solution S_x of G_x by imposing $\vec{t}$ on H_x. Indeed, consider a vertex $u \in H_x$. It may happen that u might have less than $\vec{t}(u)$ neighbors in S_y and S_z but more than $\vec{t}(u)$ in $S_y \cup S_z$. To avoid this problem, we have to consider each union $S_y \cup S_z$ for every pair $(\vec{t}_1, \vec{t}_2) \in \{1, \ldots, \text{thr}_{\max}\}^{H_x}$ such that $\vec{t}_1 + \vec{t}_2 = \vec{t}$ (where $\vec{t}_1 + \vec{t}_2$ is the classical vector component addition) and take the largest one. The final problem we have to deal with is that, whenever we make the previous union, we do not take into consideration that the sets H_y and H_z are equal. The consequence is that a vertex v in G_y might have a number of neighbors in $H_y \cap (S_y \cup S_z)$ that sums over its threshold. That's where the vector $\vec{c}$ comes into play. According to the definition of $A_x[\vec{t}, \vec{c}]$, this vector ensures that the same vertices in both H_y and H_z are in the solution. This completes the description of the algorithm, we now give the details.

Algorithm. We denote by $\vec{f} \oplus (a, b)$ the extended function $\vec{f'}$ such that $\forall x$, $\vec{f'}(x) = \vec{f}(x)$ if $x \neq a$ and $\vec{f'}(a) = b$.

Initialization step. We initialize all the tables A_x where x is a leaf of T as follows. For each leaf x of T, $\vec{t} \in \{1, \ldots, \text{thr}_{\max}\}^{H_x}$ and $\vec{c} \in \{0, 1\}^{H_x}$. Let $S = \{v \in H_x : \vec{c}(v) = 1\}$

$$A_x[\vec{t},\vec{c}] = \begin{cases} S & \text{if } S \text{ is a harmless set for } G_x \text{ according to } \vec{t} \text{ and } \vec{c} \\ \emptyset & \text{otherwise} \end{cases}$$

The running time of this step is $\text{thr}_{\max}^{|H_x|}\, 2^{|H_x|} n = O(\text{thr}_{\max}^{2\,\text{tw}}\, n)$.

Updating step. Starting from the leaves, we apply the following rules to each node $x \in X$ we visit until we reach the root.

Case 1 (insert node). Suppose that x is an insert node with child y such that $H_x = H_y \cup \{u\}$. Following the above discussion, we update the table A_x as follows. For all $\vec{t} \in \{1,\ldots,\text{thr}_{\max}\}^{H_y}$, $\vec{c} \in \{0,1\}^{H_y}$ and $i = 1,\ldots,\text{thr}_{\max}$

$$A_x[\vec{t} \oplus (u,i), \vec{c} \oplus (u,0)] = \begin{cases} A_y[\vec{t},\vec{c}] & \text{if } A_y[\vec{t},\vec{c}] \text{ is a harmless set in } G_x \\ & \text{with } \text{thr}(u) = i \\ \emptyset & \text{otherwise} \end{cases}$$

$$A_x[\vec{t} \oplus (u,i), \vec{c} \oplus (u,1)] = \begin{cases} A_y[\vec{t},\vec{c}] \cup \{u\} & \text{if } A_y[\vec{t},\vec{c}] \cup \{u\} \text{ is a harmless set} \\ & \text{in } G_x \text{ with } \text{thr}(u) = i \\ \emptyset & \text{otherwise} \end{cases}$$

The running time is $\text{thr}_{\max}^{|H_y|}\, 2^{|H_y|}\, \text{thr}_{\max}\, n = O(\text{thr}_{\max}^{2\,\text{tw}}\, n)$.

Case 2 (forget node). Suppose that x is a forget node with child y such that $H_x = H_y - \{u\}$. Let $\vec{t} \in \{1,\ldots,\text{thr}_{\max}\}^{H_y}$. Notice that vertex u has its neighbors entirely contained in G_x. Hence, the maximum harmless set for G_x where $\text{thr}(v) = \vec{t}(v)$ $\forall v \in H_x$ is exactly the maximum harmless set for G_y where $\text{thr}(v) = \vec{t}(v)$ $\forall v \in H_y$ and such that u has threshold $\text{thr}(u)$. Formally, we update the table A_x as follows. For all $\vec{t} \in \{1,\ldots,\text{thr}_{\max}\}^{H_x}$ and $\vec{c} \in \{0,1\}^{H_x}$

$$A_x[\vec{t},\vec{c}] = \max_{i \in \{0,1\}} \{A_y[\vec{t} \oplus (u,\text{thr}(u)), \vec{c} \oplus (u,i)]\}$$

The running time is $\text{thr}_{\max}^{|H_x|}\, 2^{|H_x|} n = O(\text{thr}_{\max}^{2\,\text{tw}}\, n)$.

Case 3 (join node). Suppose that x is a join node with children y and z such that $H_x = H_y = H_z = \{u_1,\ldots,u_{|H_x|}\}$. According to the above discussion, we update the table A_x as follows. For all $\vec{t} \in \{1,\ldots,\text{thr}_{\max}\}^{H_x}$ and $\vec{c} \in \{0,1\}^{H_x}$, perform the following two steps

1. $(\vec{t}_1^*, \vec{t}_2^*) = \underset{\substack{\vec{t}_1,\vec{t}_2 \in \{1,\ldots,\text{thr}_{\max}\}^{H_x} \\ :\vec{t}_1 + \vec{t}_2 = \vec{t}}}{\arg\max} |A_y[\vec{t}_1,\vec{c}] \cup A_z[\vec{t}_2,\vec{c}]|$
2. $A_x[\vec{t},\vec{c}] = A_y[\vec{t}_1^*,\vec{c}] \cup A_z[\vec{t}_2^*,\vec{c}]$

The running time of this step is $\text{thr}_{\max}^{2|H_x|}\, 2^{|H_x|} n = O(\text{thr}_{\max}^{3\,\text{tw}}\, n)$.

Final step. The optimal solution is then $\arg\max_{\vec{c} \in \{0,1\}^{H_r}} |A_r[\vec{t}_r,\vec{c}]|$ where r is the root of T and $\vec{t}_r(v) = \text{thr}(v)$ for all $v \in H_r$. ■

Using Reduction rule 40 together with Theorem 50, we immediately get the following.

Corollary 51 Harmless Set *is fixed-parameter tractable with respect to the combined parameter* k *and the treewidth* tw *of the input graph. The algorithm runs*

in $O((k+1)^{3\,\mathrm{tw}} \cdot n)$ *time.*

4.5 Approximability

In this section, we first show that Max Harmless Set is strongly inapproximable even for small constant thresholds on bipartite graphs. However, we provide a polynomial-time approximation scheme when the graph is planar. Furthermore, in the case of unanimity thresholds we prove that Max Harmless Set is APX-hard and admits a linear-time 3-approximation algorithm.

We start by showing the hardness result on bipartite graphs by an E-reduction from the Max Clique problem (see Appendix A).

Theorem 52 *For any* $\varepsilon > 0$, Max Harmless Set *with thresholds at most two cannot be approximated within* $n^{1-\varepsilon}$ *in polynomial time even on bipartite graphs, unless* NP = ZPP.

Proof. We construct an E-reduction (see Definition 31) from Max Clique. Let G be an instance of Max Clique. Consider the constructed instance $I' = (G', \mathrm{thr})$ from G as it is defined in Theorem 44. Let C be a harmless set in G'. From the proof of Theorem 44, we know that C is a clique in G. Thus, it is not hard to see that $\mathrm{opt}(I') = \mathrm{opt}(G)$ and $\varepsilon(G, C) = \varepsilon(I', C)$. Since Max Clique is not approximable within $n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless NP = ZPP [66], the result follows. ■

In the following we propose a polynomial-time approximation scheme on the class of planar graphs, using the previous polynomial-time algorithm for graphs of bounded treewidth.

Theorem 53 Max Harmless Set *is in* PTAS *on planar graphs.*

Proof. Let $I = (G, \mathrm{thr})$ be an instance of Max Harmless Set. Given a planar embedding of an input graph, we consider the set of the vertices which are on the exterior face, they will be called *level 1 vertices*. By induction we define *level* k as the vertices which are on the exterior face when we have removed the vertices of levels smaller than k [13]. A planar embedding is *k-level* if it has no nodes of level greater than k. If a planar graph is k-level, it has a k-outerplanar embedding.

If we want to achieve an approximation within $1+\varepsilon$, let us consider $k = 2(1+\lceil \frac{1}{\varepsilon} \rceil)$. Let X_t be the set of vertices of level t and let H_i, $0 \leq i \leq k-1$, be the graph obtained from G by considering the subgraphs formed by the set of vertices $\bigcup_{t+1 \leq j \leq t+k} X_j$, for $t \equiv i \pmod{(k-2)}$. The subgraph containing exactly $\bigcup_{t+1 \leq j \leq t+k} X_j$ is k-outerplanar, and so is H_i, too.

Since H_i is k-outerplanar, it has treewidth at most $3k-1$ [19]. We construct graph H_i' from H_i by attaching a forbidden edge to each vertex on the boundary (that means vertices in $X_{t+1}, X_{t+2}, X_{t+k-1} X_{t+k}$ with $t \equiv i \pmod{(k-2)}$). Thus, in each subgraph of H_i' the vertices in $X_{t+1}, X_{t+2}, X_{t+k-1} X_{t+k}$ cannot take part from any harmless set.

On applying Theorem 50, we can efficiently determine an optimal harmless set in each subgraph of H_i'. Denote by S_i the union of these harmless sets. Clearly S_i is a harmless set on H_i.

Among $S_0, \ldots, S_{k-1}$ we choose the best solution that we denote S and we are going to prove that S is an $(1+\varepsilon)$-approximation of the optimal value on G. We can easily show that there is at least one r, $0 \leq r \leq k-1$ such that at most $\frac{2}{k}$

of vertices in an optimal solution S_{opt} of G are on levels $X_{t+1}, X_{t+2}, X_{t+k-1}X_{t+k}$ with $t \equiv r \pmod{(k-2)}$). This means that the solution S_r obtained by deleting the vertices from levels $X_{t+1}, X_{t+2}, X_{t+k-1}X_{t+k}$ from S_{opt} will have at least $|S_{opt}|(1-\frac{2}{k}) = \frac{k-2}{k}\operatorname{opt}(I)$ vertices. According to our algorithm, $|S| \geq |S_r| \geq \frac{\operatorname{opt}(I)}{1+\varepsilon}$.

The overall running time of the algorithm is k times what we need for graphs of treewidth at most k, that is $O(k \operatorname{thr}_{\max}^{6k-2} n) = n^{O(1/\varepsilon)}$ where $\operatorname{thr}_{\max}$ is the maximum threshold. ■

Unanimity thresholds. As previously observed in the section dedicated the parameterized complexity of Harmless Set, it seems like unanimity thresholds make the problem more "tractable". We confirm this fact from an approximation perspective by showing the APX-completeness of Max Harmless Set with unanimity thresholds.

Lemma 54 Max Harmless Set *with unanimity thresholds is* 3*-approximable in linear time.*

Proof. Let I be an instance of Max Harmless Set consisting of a graph G and unanimity thresholds. The algorithm consists of the following two steps:

1. Compute a spanning tree T of G.
2. Compute an optimal solution S of T using Theorem 49 with unanimity threshold.

Observe that any feasible solution S for T is also a solution for G. Indeed, if a vertex v in T is such that $N_T(v) \not\subseteq S$ then we have $N_G(v) \not\subseteq S$. Moreover, using Theorem 45, we get $|S| \geq n/3 \geq \operatorname{opt}(I)/3$. ■

Theorem 55 Max Harmless Set *with unanimity thresholds is* APX*-complete.*

Proof. Membership follows from Lemma 54. In order to prove the APX-hardness we provide an L-reduction (see Definition 30) from the APX-hard problem Max E2Sat-3 (see Appendix A).

Given a formula ϕ of Max E2Sat-3 with n variables and m clauses (notice that $m = 3n/2$), we construct an instance I of Max Harmless Set consisting of a graph $G = (V, E)$ and unanimity thresholds as follows (see Figure 4.2). For every variable x_i, we construct the complete bipartite graph $K_{3,3}(x_i) = (V^-(x_i), V^+(x_i))$ in which every edge uv is replaced by an edge-vertex e_{uv} and two edges ue_{uv} and $e_{uv}v$. We denote by $E(x_i)$ this set of edge-vertices. The vertices in $V^+(x_i)$ (resp. $V^-(x_i)$) represents the positive (resp. negative) literals of x_i. We denote by A the set of all vertices added so far. For every clause c_j in ϕ add two adjacent clause-vertices $\bar{c}_j$ and $\bar{c}'_j$. For every variable x_i, if x_i appears positively (resp. negatively) in a clause c_j then add an edge between $\bar{c}'_j$ and a vertex of $V^-(x_i)$ (resp. $V^+(x_i)$). Thus, vertex $\bar{c}_j$ represents the complement of the clause c_j in ϕ. Finally, add two adjacent vertices c and c'. For every vertex $v \in V^-(x_i) \cup V^+(x_i)$, if v is not adjacent to a clause-vertex then add the edge vc'. This completes the construction.

The optimal value in I is bounded by the number of vertices of G and thus, $\operatorname{opt}(I) \leq 15n + 2m + 2 \leq 16\operatorname{opt}(\phi) + 2 \leq 18\operatorname{opt}(\phi)$ since $\operatorname{opt}(\phi) \geq 3/4m$ and $\operatorname{opt}(\phi) \geq 1$.

Moreover, let x^* be an optimal assignment for ϕ and let

$$S = \bigcup_{i:x_i^*=1} V^+(x_i) \cup \bigcup_{i:x_i^*=0} V^-(x_i) \cup \bigcup_{i=1}^{n} E(x_i) \cup \{\bar{c}_j : c_j \text{ is satisfied by } x^*\} \cup \{c\}.$$

We can easily verify that S is a harmless set and $|S \cap (V^-(x_i) \cup V^+(x_i) \cup E(x_i)| = 12$ and thus $|S \cap A| = 8m$ and then $\text{opt}(I) \geq |S| = 8m + \text{opt}(\phi) + 1$.

Let S be a harmless set for I. We show in the following how to construct an assignment a_S for ϕ from the solution S such that $\text{cost}(\phi, a_S) = |S| - 8m - 1$. For each variable x_i, S cannot contain vertices from both $V^-(x_i)$ and $V^+(x_i)$ since otherwise an edge-vertex has both neighbors inside S. Notice also that S cannot contain any vertex $\bar{c}'_j$, since $\bar{c}'_j$ is adjacent to the degree one vertex $\bar{c}_j$. Similarly $c' \notin S$.

If S contains for every $i = 1, \ldots, n$ the set $E(x_i)$ and one of the sets $V^-(x_i)$ or $V^+(x_i)$ then $|S \cap A| = 8m$ and we can defined the following assignment a_S: $x_i = 1 \Leftrightarrow |S \cap V^+(x_i)| \neq 0$. In this case, a clause-vertex is in S if and only if the corresponding clause is satisfied by a_S. Thus, the number of clauses satisfied by a_S is exactly $\text{cost}(\phi, a_S) = |S| - 8m - 1$.

Assume now that $|S \cap A| < 8m$. We show that there exists an other solution S' with $|S'| \geq |S|$ such that $|S' \cap A| = 8m$. If a vertex $v \in E(x_i) \setminus S$ for some $i \in \{1, \ldots, n\}$, we can add v in S since v cannot have both neighbors in S. Similarly, if c is not in S, then we add c in S. Furthermore, we may assume that for each $i = 1, \ldots, n$, we have either $|V^-(x_i) \cap S| \geq 1$ or $|V^+(x_i) \cap S| \geq 1$. Indeed, there is always a vertex $v \in V^-(x_i) \cup V^+(x_i) \cap N(c')$ that can be added to S since $c' \notin S$. Moreover, we have $|V^-(x_i) \cap S| < 3$ and $|V^+(x_i) \cap S| < 3$ for some $i \in \{1, \ldots, n\}$ since $|S \cap A| < 8m$. Let $i \in \{1, \ldots, n\}$ be such that $1 \leq |V^-(x_i) \cap S| \leq 3$ (the case $1 \leq |V^+(x_i) \cap S| \leq 3$ is symmetric). There must exist a vertex $v \in V^-(x_i) \setminus S$ which is either adjacent to c' or to a clause-vertex $\bar{c}'_j$. In the first case we add v in S. In the second case, we denote $N(\bar{c}'_j) = \{v, v', \bar{c}_j\}$. If $v' \in S$ and $\bar{c}_j \in S$ then remove $\bar{c}_j$ from S and add v instead, otherwise add v in S. Thus, we obtain a new solution S' such that $|S'| \geq |S|$ and $|S' \cap A| = 8m$. Similarly to the above case, we can obtain an assignment $a_{S'}$ such that $|S'| - \text{cost}(\phi, a_{S'}) = 8m + 1$. In particular, if S' is an optimal solution, then $\text{opt}(\phi) \geq \text{cost}(\phi, a_{S'}) = \text{opt}(I) - 8m - 1$ and thus, we have $\text{opt}(I) - \text{opt}(\phi) = 8m + 1$ and then $\text{opt}(\phi) - \text{cost}(\phi, a_{S'}) = \text{opt}(I) - |S'|$. ■

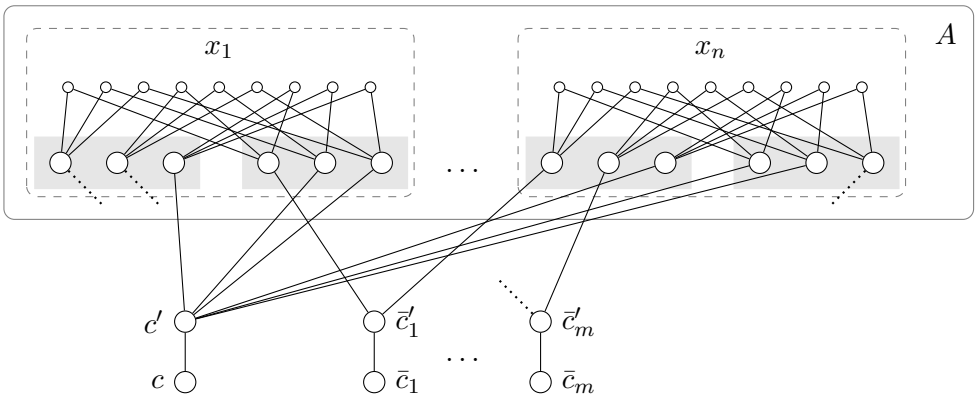


Figure 4.2: The construction of G. The subgraph in a dashed box corresponds to a gadget $K_{3,3}(x_i)$ of a variable x_i. A grey box represents either $V^+(x_i)$ or $V^-(x_i)$.

4.6 Conclusion and open problems

In this chapter, we introduced the harmless set problem. We established positive and negative results concerning its parameterized tractability and polynomial-time approximability. However, several questions remain open.

(1) Is Harmless Set fixed-parameter tractable for the parameter treewidth?

(2) We showed that there exists a ptas for Max Harmless Set on planar graphs. However, the parameterized complexity of Harmless Set parameterized by k in this class of graphs is open.

(3) There is room enough for improving the approximability of Max Harmless Set with unanimity thresholds.

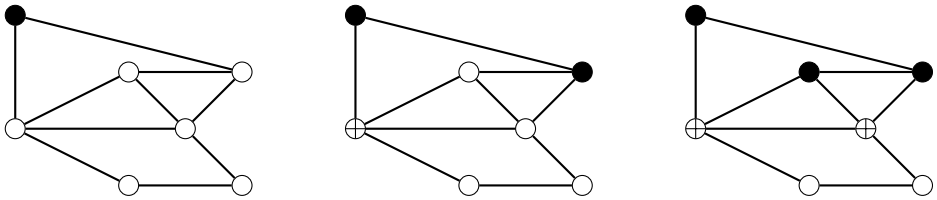
(4) Another interesting open question is whether Harmless Set with unanimity thresholds is fixed-parameter tractable with respect to the parameter k when we ask to determine the existence of a harmless set of size at least $\lceil \frac{n}{3} \rceil + k$.

(5) The parameterized approximability of the problem has not been studied and thus, left as an open question.

Chapter

5 Containing an undesirable spread

Following the idea of preventing an undesirable thing to propagate through a network, we investigate the complexity of the firefighter problem. This problem has been introduced by Hartnell in 1995 [64] and concerns a deterministic model of "fire" spreading through a graph via its edges. Of course, the term fire can be replaced by any other relevant term depending of the application context (*e.g.* rumor, virus, ideas, etc.). Formally, it consists of a graph with an initially burned vertex. At each time step, we are first allowed to protect one vertex and then the fire spreads to every unprotected vertex adjacent to the fire. When no new vertex can burn, the process stops. The goal is to minimize the number of burned vertices. This can be seen as a "dynamic" cut problem since protecting a vertex has the same effect as removing it from the graph. In the figure below, we depict an example of such process. In the figure, the burned vertices are represented by black vertices. A protected vertex is represented by a circle with a cross inside.



Step 1 Step 2 Step 3

In this chapter, we study the classical complexity, parameterized complexity and approximability of a generalized version of this problem where $b > 1$ vertices can be protected at each step as well as some variants.

The content of this chapter is based on the following papers.

▶ C. Bazgan, M. Chopin and M. R. Fellows, *Parameterized complexity of the firefighter problem*, Proceedings of the 22nd International Symposium on Algorithms and Computation (ISAAC 2011), LNCS 7074, pp. 643–652, 2011.

▶ C. Bazgan, M. Chopin and B. Ries, *The firefighter problem with more than one firefighter on trees*, Discrete Applied Mathematics 161(7-8), 899-908, 2013.

▶ C. Bazgan, M. Chopin, M. R. Fellows, F. V. Fomin, E. J. van Leeuwen and M. Cygan, *Parameterized Complexity of Firefighting*, submitted.

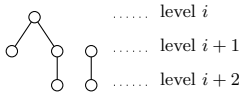
▶ J. Chlebíková and M. Chopin, *The Firefighter Problem: A Structural Analysis*, ongoing paper.

5.1 Introduction

In the Chapter 3, we analyzed the computational complexity of maximizing the spread of information through a network. Motivated by applications such as the containment of a disease, we may modify the propagation process defined in that chapter by allowing the possibility of "vaccinate" some vertices. Once a vertex has been vaccinated, it can no longer be activated. Different objectives may then be of interest, for instance minimizing the total number of activated vertices by vaccinating some particular vertices, or making sure that some specific subset of vertices does not get activated at all, etc. In the literature, these questions have been referred to as the firefighter problem. In the context of this problem and, consequently, throughout this chapter, an activated vertex will be called a burned vertex and a vaccinated vertex is said to be protected. The firefighter problem was introduced by Hartnell [64] and further received considerable attention [7, 27, 45, 55, 64, 65, 69, 75, 82, 89]. Originally, the problem was defined as follows. Initially, a fire breaks out at some vertex s of a graph. At each time step, we have to choose one vertex which will be protected by a firefighter. Then the fire spreads to all unprotected neighbors of the vertices on fire. The process ends when the fire can no longer spread, and then all vertices that are not on fire are considered as saved. The objective consists of choosing, at each time step, a vertex which will be protected by a firefighter such that a maximum number of vertices in the graph is saved at the end of the process.

The firefighter problem was proved to be NP-hard for bipartite graphs [82], cubic graphs [75] and unit disk graphs [58]. Finbow et al. [55] showed that the problem is NP-hard even on trees. More precisely, they proved the following dichotomy theorem: the firefighter problem is NP-hard even for trees of maximum degree three and it is solvable in polynomial-time for graphs with maximum degree three, provided that the fire breaks out at a vertex of degree at most two. Furthermore, the firefighter problem is polynomial-time solvable for caterpillars and P-trees [82].[1] From the approximation point of view, the problem is $\frac{e}{e-1}$-approximable on trees ($\frac{e}{e-1} \approx 1.5819$) [27] and it is not $n^{1-\varepsilon}$-approximable on general graphs for any $\varepsilon > 0$, if $\mathsf{P} \neq \mathsf{NP}$ [7]. Moreover for trees where vertices have at most three children, the firefighter problem is 1.3997-approximable [69]. Finally, Cai et al. [27] gave fixed-parameter tractable algorithms and polynomial-size kernels for trees for each of the following parameters: "number of saved leaves", "number of burned vertices", and "number of protected vertices". The problem on grid graphs of dimension two and

[1] A P-tree [82] is a tree which does not contain the following configuration.



higher has been particularly investigated [45, 89, 82, 102]. There are also several variants of the firefighter problem. For instance, Anshelevich et al. [7] considered the "spreading vaccination model" where both the fire and the firefighters propagate through the network. In this case, the problem admits a polynomial-time 2-approximation algorithm and a randomized $\frac{e}{e-1}$-approximation algorithm. The politician's firefighter, introduced by Scott et al. [99], is a more "localized" version: At each time step, we are allowed to put as many firefighters as the number of burned vertices — we invest more ressources as the situation gets more threatening. Moreover, these firefighters can only be placed next to burned vertices — one would prefer to fight the threat closely than miles away. On the contrary to the classical problem, this variant is polynomial-time solvable on trees and fixed-parameter tractable for general graphs when parameterized by the "number of burned vertices" [99]. However, it has been shown that the problem remains NP-hard even for planar graphs [99]. Another problem related to the firefighter problem was introduced by King and MacGillivray [75]. It consists of deciding whether there is a strategy of choosing a vertex to be protected at each time step such that all vertices of a given set S are saved. Similarly to the classical problem, it has been proved NP-complete even for trees of maximum degree three in which every leaf is at the same distance from the vertex where the fire starts and S is the set of leaves. However, it is polynomial-time solvable for trees of maximum degree three if the initially burned vertex has degree at most two. For a more complete overview and open questions about the firefighter problem, the reader is referred to the recent survey of Finbow and MacGillivray [54]. In this thesis, we consider the version of the firefighter problem which allows us to protect $b \geq 1$ vertices at each step (the value b is called *budget*). We then denote by Firefighter the decision problem that asks for the existence of a protection strategy with respect to a budget $b > 0$ such that at least k vertices are saved. We also consider its parametric dual Dual Firefighter that asks to save at least $n - k_b$ vertices for some integer k_b, and the Bounded Firefighter problem as well. This last problem is defined similarly to Firefighter except that we are allowed to protect a total of at most $k_p \geq 1$ vertices where k_p is an integer given in the input. We are also interested in the generalized version of the Save problem introduced by King and MacGillivray [75] that asks for the existence of a protection strategy with respect to a budget $b > 0$ such that no vertex of a given vertex subset is burned.

This chapter is organized as follows. First, we provide the formal definitions of the previous problems in Section 5.2 as well as some preliminaries. In Section 5.3, we show that the Firefighter problem is NP-complete for trees of bounded degree $b+3$ and for trees of bounded pathwidth three for any fixed budget $b \geq 1$. However, we prove that the problem is polynomial-time solvable for the class of graphs of both bounded degree and bounded pathwidth, and thus for graphs of bounded bandwidth. We also design a polynomial-time algorithm for solving the problem (and the corresponding weighted version) on a subclass of trees of pathwidth two, namely k-caterpillars. In Section 5.4, we establish the parameterized complexity lower and upper bounds of the problems Firefighter, Dual Firefighter, and Bounded Firefighter on general graphs with respect to their respective standard parameterization (see Figure 5.2). In Section 5.5, we give parameterized algorithms that answer several open questions from Cai et al. [27] and refine some of their results as well. In Section 5.6, we provide other positive parameterized complexity results with respect to the parameter vertex cover (see Figure 5.1). In Section 5.7, we observe that the minimum version of the Firefighter problem where the goal is to minimize the number of burned vertices is not $n^{1-\varepsilon}$-approximable on trees for any $\varepsilon > 0$ and any $b \geq 1$ if P $\neq$NP. We also answer negatively an open question of Finbow and MacGillivray [54]. Conclusion and open problems are given in Section 5.8.

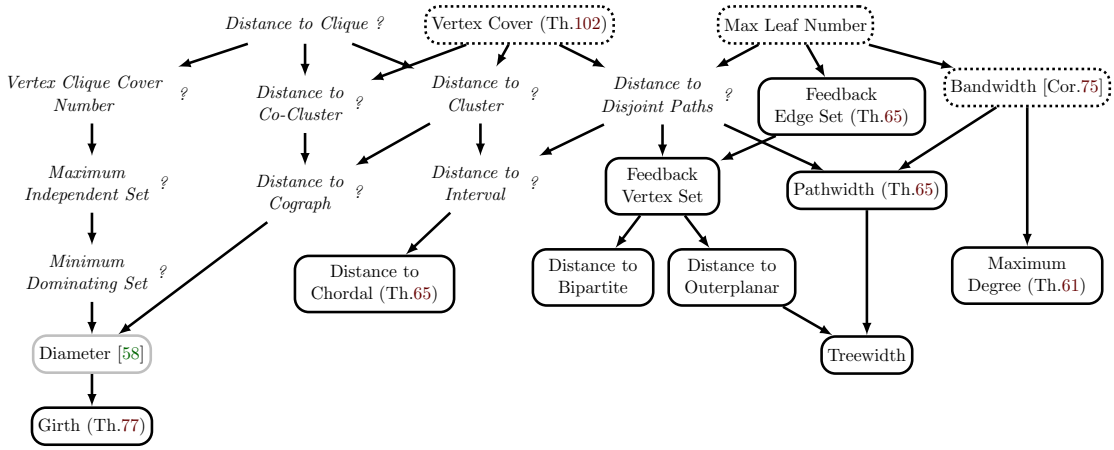


Figure 5.1: The parameterized complexity of the FIREFIGHTER problem with respect to some structural graph parameters (see Section 2.3.4). A dotted rectangle means FPT for this parameter, a gray rectangle indicates a W[1]-hardness result for any fixed budget, and a thick rectangle means that the problem is NP-hard for constant values of this parameter and any fixed budget. A parameter written in italic followed by a question mark indicates an open question. For the parameter "diameter", Fomin et al. [58] showed that FIREFIGHTER is in XP.

	FIREFIGHTER k	BOUNDED FIREFIGHTER k_p	DUAL FIREFIGHTER k_b
	W[1]-hard	W[1]-hard	W[1]-hard
Poly Kernel?	no	no	no
Budget	W[1]-hard	W[1]-hard	FPT
Poly Kernel?	no	no	no
Treewidth	FPT	FPT	?
Poly Kernel?	?	no	no

Figure 5.2: Summary of our parameterized complexity results including standard parameterizations. Each column is associated to a problem with its standard parameter and each line corresponds to a parameter as well (see the next section for the problems definition). The intersection of a line with a column corresponds to a parameterized complexity result with combination of the line's parameter and column's parameter. The first line corresponds to parameterized complexity results solely based on the standard parameters.

5.2 Problem definitions and preliminaries

Before defining the problems, we need to explain the spreading process at stake. Let $G = (V, E)$ be a graph of order n with a vertex $s \in V$ and $b > 0$ be an integer called *budget*. At step $t = 0$, a fire breaks out at vertex s and s starts burning. At any subsequent step $t > 0$, the following two phases are performed in sequence:

1. *Protection phase* : Protect (or defend) at most b vertices not yet on fire.

2. *Spreading phase* : Every unprotected vertex which is adjacent to a burned vertex starts burning.

Burned and protected vertices remain burned and protected until the process stops, respectively. The process stops when in a step no new vertex can be burned. We call a vertex saved if it is either protected or if all paths from any burned vertex to it contains at least one protected vertex. Notice that, until the propagation process stops, there is at least one new burned vertex at each step. This leads to this obvious lemma.

Lemma 56 *The number of steps before the propagation process stops is less or equal to the total number of burned vertices.*

A *protection strategy* (or simply *strategy*) indicates which vertices to protect at each step until the propagation process stops. Since there can be at most n burned vertices, it follows from Lemma 56 that the propagation unfolds in at most n steps. We thus denote a strategy as a sequence $\Phi = (D_1, D_2, \ldots, D_n)$ where $D_i \subseteq V$ and $|D_i| \leq b$ for every $i = 1, \ldots, n$. Each set D_i contains the vertices of G defended at step i.

In an algorithmic perspective, it is more convenient to use the following equivalent protection phase definition:

Constrained protection phase : Protect (or defend) vertices not yet on fire under the following restrictions:

1. Each protected vertex must have a neighbor which is on fire.

2. After i steps of propagation at most ib vertices are protected.

The following lemma shows the equivalence between the original protection phase and the constrained protection phase.

Lemma 57 *For a given strategy complying with the original protection phase, there exists a strategy that respects the constrained protection phase and saves exactly the same set of vertices. The converse is true.*

Proof. Let D be the set of protected vertices of a strategy Φ complying with the original protection phase. We construct a new strategy Φ' that protects, during the i^{th} step, exactly those vertices of D which have a neighbor on fire. Clearly after i steps at most ib vertices will be protected, since each vertex of D is protected by the strategy Φ' not earlier than by the strategy Φ. Thus Φ' respects the rules of the *constrained* protection phase. Moreover, it saves exactly the same set of vertices. Conversely, let D be the set of protected vertices of a strategy Φ' complying with the constrained protection phase. We construct a strategy Φ as follows. Let $(v_1, \ldots, v_{|D|})$ be a sequence of vertices of D sorted by the step in which a vertex is protected by Φ' (breaking ties arbitrarily). In the i-th step of strategy Φ, we protect the vertices $v_{(i-1)b+1}, \ldots, v_{ib}$. The vertex v_j, $j \in \{(i-1)b+1, \ldots, ib\}$, is not on fire in the i^{th} step, because in the strategy Φ' it is protected not earlier than in the i-th step. Thus Φ respects the rules of the original protection phase and saves exactly the same set of vertices. ■

Unless otherwise stated, we assume to use the original protection phase. We are now in position to give the formal definitions of the investigated problems.

FIREFIGHTER
Input: A graph $G = (V, E)$, a vertex $s \in V$, and two integers b and k.
Question: Is there a strategy with respect to budget b such that at least k vertices are saved when a fire breaks out at s?

In the above problem, we may assume without loss of generality that $b < k$ since otherwise the problem is trivial: At time step one, if it is possible to protect $b \geq k$ vertices then the answer is "yes", otherwise the answer is "no". The parametric dual, denoted by DUAL FIREFIGHTER, asks for the existence of a strategy such that at least $n - k_b$ vertices are saved or, equivalently, at most k_b vertices are burned. We denote by MAX FIREFIGHTER the maximization version where the goal is to save the maximum number of vertices. The minimization version will be denoted by MIN FIREFIGHTER and consists in finding a strategy that minimizes the number of burned vertices.

We also studied the following variant where the total number of protected vertices is bounded.

BOUNDED FIREFIGHTER
Input: A graph $G = (V, E)$, a vertex $s \in V$, and three integers k, b and k_p.
Question: Is there a strategy with respect to the budget b that saves at least k vertices by protecting a total of at most k_p vertices when a fire breaks out at s?

The corresponding maximization version, denoted by MAX BOUNDED FIREFIGHTER, asks for finding a strategy that saves the maximum number of vertices by protecting a total of at most k_p vertices. We may assume without loss of generality that $b \leq k_p$ in any instance of BOUNDED FIREFIGHTER and MAX BOUNDED FIREFIGHTER. Indeed, having a budget $b \geq k_p$ is useless since we can protect a total of at most k_p vertices. In other words, the answer of any instance with a budget $b \geq k_p$ does not change if we set $b = k_p$.

We investigate moreover the following generalized version of the problem introduced by King and MacGillivray [75].

SAVE
Input: A graph $G = (V, E)$, a "critical" set $C \subseteq V$, a vertex $s \in V$, and an integer b.
Question: Is there a strategy with respect to budget b saving all the vertices that belong to C when a fire breaks out at s?

It is worth noting that the above problem was originally entitled "FIRE" [75]. However, we find the name "SAVE" more relevant regarding the objective of the problem.

Finally, suppose that we are given a weight $w(v)$ for each vertex v of a graph G. These weights may for instance reflect the importance of the vertices: if $w(v_1) > w(v_2)$, vertex v_1 is considered as more important than vertex v_2. Then we may define the following weighted version of the MAX FIREFIGHTER problem.

MAX WEIGHTED FIREFIGHTER
Input: A graph $G = (V, E)$, a vertex $s \in V$, a weight function $w : V \to \mathbb{N}$, and an integer b.
Output: A strategy with respect to budget b that maximizes the total weight of the saved vertices in G when a fire breaks out at s.

When the weights are either 0 or 1, we denote the Max Weighted Firefighter problem as the Max 0-1 Firefighter problem.

When dealing with trees, we use the following observation which is a straightforward adaptation of the one by MacGillivray and Wang for the case $b > 1$ [82, Section 4.1].

Lemma 58 *Among the strategies that maximizes the number (resp. the total weight) of saved vertices for a tree, there exists one that protects vertices adjacent to a burned vertex at each time step.*

In other words, the Lemma 58 implies that we can only consider strategies that protect vertices adjacent to a burned vertex at each time step on a tree.

It is worth pointing out that the previous lemma does not hold for general graphs. Indeed, an optimal solution might protect some vertex away from the fire at some step (see for instance Figure 5.3).

Figure 5.3: In the figure (a) the applied strategy consists of protecting a vertex adjacent to the fire at each step. The strategy of figure (b) protects vertex v_3 at time step one and v_4 at time step two. The later strategy save one more vertex than the former one. The burned vertices are represented by black vertices. A protected vertex is represented by a circle with a cross inside.

5.3 The importance of bounded degree and "path-likeness"

In this section, we emphasize the importance of the parameters pathwidth and maximum degree in the computational complexity of the Firefighter problem. More precisely, we prove that if both are bounded then the problem is polynomial-time solvable, but it is NP-complete if only one of these two parameters is bounded.

5.3.1 Graphs of bounded degree

It has been shown in [55] that Save is NP-complete for trees of maximum degree three when one firefighter is available at each step (*i.e.* $b = 1$). Furthermore, it is polynomial-time solvable for trees of maximum degree three if the fire breaks out at a vertex of maximum degree two. In the following, we extend these results for any fixed budget $b \geq 1$. We then rely on these last results to show the NP-completeness of Firefighter on trees of bounded degree.

Maximum degree $b + 3$. We first show the NP-completeness of Save on trees for any fixed budget $b \geq 1$. Before doing that, we need the following lemma.

Lemma 59 *Let T be a tree with a burning root s together with a strategy that saves all the leaves of T. Suppose that there exist levels i and $j > i$ containing $b_i \leq b-1$, for some budget $b \geq 1$, and $b_j \geq 1$ firefighters, respectively. Then there exists a new strategy saving all the leaves of T and such that levels i and j contain b_i+1 and b_j-1 firefighters, respectively.*

Proof. Let v_j be a protected vertex by the strategy at a level $j > i$, and let v_i be the ancestor of v_j at level i. It follows from Lemma 58 that we may assume that v_i is not protected. We transform this strategy into a new one as follows (see Figure 5.4): protect v_i at time step i and do not protect v_j at time step j. Since v_i is an ancestor of v_j, it follows that using the new strategy, we save a subset of vertices that contains the vertices saved by using the former strategy. Since level i contains at most $b-1$ firefighters it follows that the new strategy is valid, saves all the leaves of T and levels i and j contain respectively b_i+1 and b_j-1 firefighters. ■

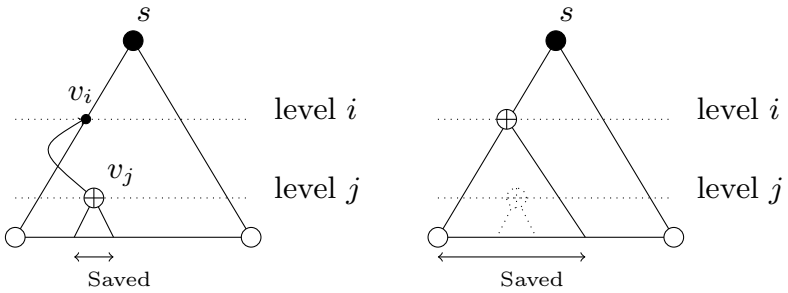


Figure 5.4: Moving up a firefighter leads to a strategy that saves at least the same set of leaves.

Theorem 60 *For any fixed budget $b \geq 1$ and any critical set,* Save *is* NP*-complete for trees of maximum degree $b+2$.*

Proof. Clearly, Save belongs to NP. In order to prove its NP-hardness for any fixed budget $b \geq 1$, we reduce in polynomial time every instance of Save with budget b to a new instance of the same problem with budget $b+1$. Since Save is NP-hard for $b = 1$ [75], it follows that Save is NP-complete for any fixed $b \geq 1$.

Let $I = (T, C, s, b)$ be an instance of Save where T is a tree rooted at s of maximum degree $b+2$, height h, and size n. We may assume without loss of generality that C is the set of leaves of T. Otherwise for each non-leaf vertex $v \in C$, since we have to save v, we can remove the subtree rooted at v such that v becomes a leaf. We construct a new instance $I' = (T' = (V', E'), C', s', b+1)$ of the same problem consisting of a tree T' of maximum degree $b+3$ rooted at vertex s', and the critical set C' which corresponds to the leaves of T' as follows (see Figure 5.5).

- Add a vertex s'.
- Add two paths $\{y_1y_2, \ldots, y_{h-2}y_{h-1}\}$, $\{x_1x_2, \ldots, x_{h-1}x_h\}$, make y_1, x_1 adjacent to s' and make y_{h-1} adjacent to s.
- Add vertices $v_1, \ldots, v_{b+1}$ and make them adjacent to s'.

- For every vertex y_i, $i = 1 \ldots, h-1$, add vertices $v_{i,1}, \ldots, v_{i,b+1}$ and make them adjacent to y_i.
- For $i = 1, \ldots, h$ add a path $\{w_{i,1}w_{i,2}, \ldots, w_{i,h-1}w_{i,h}\}$ and make $w_{i,1}$ adjacent to x_i.

This clearly gives us a tree T' of maximum degree $b+3$ rooted at vertex s' and the set of leaves $C' \subset V'$ is given by $C' = C \cup \bigcup_{i=1}^{h-1}\{v_{i,1}, \ldots, v_{i,b+1}\} \cup \{w_{1,h}, \ldots, w_{h,h}\} \cup \{v_1, \ldots, v_{b+1}\}$.

We claim that there exists a strategy for I that saves all the vertices in C if and only if there exists a strategy for I' that saves all the vertices in C'.

"$\Rightarrow$": Suppose there exists a strategy $\Phi = (D_1, D_2, \ldots, D_n)$ for I that saves all the vertices in C. In order to save all vertices in C', we will apply the strategy defined as follows: at time step $t = 1$, we have to protect the vertices $v_1, \ldots, v_{b+1}$; at each time step $2 \leq t \leq h$, we have to protect the vertices $v_{t-1,1}, \ldots, v_{t-1,b+1}$; thus after time step h, vertex s is burning as well as vertices $w_{1,h-1}, w_{2,h-2}, \ldots, w_{h-1,1}, x_h$; at each time step $h+1 \leq t \leq 2h$, we protect the vertices in T belonging to D_{t-h} and we use the additional firefighter to protect the leaf $w_{t-h,h}$. This clearly gives us a valid strategy saving all the vertices in C'.

"$\Leftarrow$": Suppose now that there exists a strategy Φ' for I' that saves all the vertices in C'. At time step $t = 1$, this strategy necessarily consists in protecting vertices $v_1, \ldots, v_{b+1}$. Furthermore, at each time step $2 \leq t \leq h$, we have to protect the vertices $v_{t-1,1}, \ldots, v_{t-1,b+1}$. It follows from Lemma 58 that we may assume that Φ' is a strategy which, at each time step, protects vertices adjacent to burning vertices. Thus Φ' protects, at each time step i, at most $b+1$ vertices at level i in T' for $i = h+1, \ldots, 2h$. Let $b_T(i)$ be the number of firefighters in the subtree T of T' at level i used by Φ' and let $\mathcal{B}_T = \{i : b_T(i) = b+1\}$. If $\mathcal{B}_T = \emptyset$, then for any i, $b_T(i) \leq b$ and thus the strategy Φ', restricted to the tree T, is a valid strategy for I that saves all the leaves of T. So we may assume now that $\mathcal{B}_T \neq \emptyset$.

Let i^ℓ be the ℓ^{th} smallest value in $\mathcal{B}_T$. Consider the case $\ell = 1$. Suppose that for any $i < i^1$, $b_T(i) \geq b$. From the definition of i^ℓ, it follows that we cannot have $b_T(i) = b+1$, thus $b_T(i) = b$ for every $i < i^1$. By construction, this means that, at each time step $i < i^1$, the additional firefighter protects the vertex $w_{i-h,h}$, $i \geq h+1$. At time step i^1, since $b_T(i^1) = b+1$, the vertex $w_{i^1-h,h}$ is not protected and burns which is a contradiction. Thus there exists a level $i < i_1$ such that $b_T(i) < b$. It follows from Lemma 59 that there exists a strategy saving the leaves of T' such that $b_T(i) \leq b$ and $b_T(i^1) = b$. Applying this argument iteratively for $i^2, \ldots, i^{|\mathcal{B}_T|}$, we obtain a new strategy Φ'' that saves all the vertices in C' and such that for any level i, $b_T(i) \leq b$. Thus, the strategy Φ'' restricted to the tree T is a valid strategy that saves all the leaves of T. This completes the proof. ■

In what follows, we use the previous the result to show the NP-completeness of Firefighter on trees of bounded degree.

Theorem 61 *For any fixed budget $b \geq 1$,* Firefighter *is* NP*-complete for trees of maximum degree $b+3$.*

Proof. We construct a polynomial-time reduction from Save to Firefighter. In this reduction, we will make use of the following gadget. We denote by $\mathcal{T}(r, h, d)$ a complete tree of height h and root r such that every non-leaf vertex has exactly d children and every leaf is at the same distance from the root. For such a tree we obtain

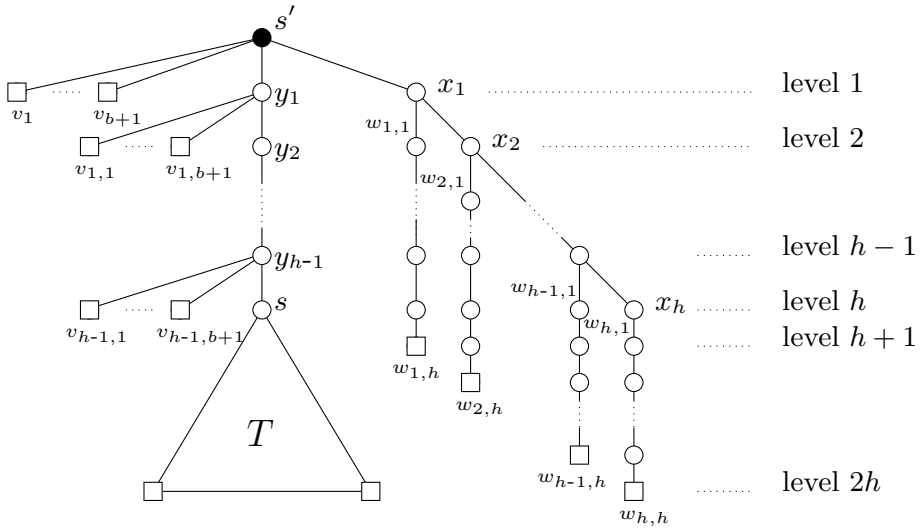


Figure 5.5: The construction of T'. The vertices from C are represented by a square.

the following property. Consider a complete tree $\mathcal{T}(r, h, b+1)$. If the fire breaks out at r, then at least one leaf will not be saved. Indeed, since each non-leaf vertex has exactly $b+1$ children, it follows that at each time step there will be at least one new burning vertex. Thus at the end of the process, at least one leaf will be burned.

Let I be an instance of Save consisting of a tree $T = (V, E)$ of maximum degree $b+2$ with $|V| = n$, a burned vertex $s \in V$, and a subset $C \subseteq V$ which corresponds to the set of leaves. We construct an instance I' of Firefighter consisting of a tree $T' = (V', E')$, and a positive integer k as follows (see Figure 5.6). For every leaf ℓ of T, add $b+2$ copies $\mathcal{T}_{1,\ell}, \ldots, \mathcal{T}_{b+2,\ell}$ of the tree $\mathcal{T}(r, \lceil \log_{b+1} n + 1 \rceil, b+1)$ such that the root $r_{i,\ell}$ of $\mathcal{T}_{i,\ell}$ is adjacent to ℓ, for $i \in \{1, \ldots, b+2\}$. Let $|\mathcal{T}|$ denote the cardinality of each of those trees. Notice that each tree $\mathcal{T}_{i,\ell}$ has $|\mathcal{T}| \geq n$ vertices. Set $k = (b+2)|C||\mathcal{T}|$.

We will prove that there exists a strategy for I that saves all the vertices in C if and only if there exists a strategy for I' that saves at least k vertices in C'.

"$\Rightarrow$": Suppose there exists a strategy for I that saves all the vertices in C. Since C is the set of all leaves in T, it follows that this strategy applied to T' saves all the vertices of the trees $\mathcal{T}_{i,\ell}$. Notice that we have $(b+2)|C|$ such trees. Thus the number of saved vertices in T' is at least $k = (b+2)|C||\mathcal{T}|$.

"$\Leftarrow$": Conversely, suppose that no strategy for I can save all the vertices in C. Thus, at least one leaf of T is burned at the end. This necessarily implies that for any strategy Φ' for I' there is at least one vertex, say ℓ, of C which is burned. It follows from the construction of T' and the property of each tree $\mathcal{T}_{i,\ell}$, that in this case there are at least $|\mathcal{T}|$ vertices which will be burned for strategy Φ'. Thus Φ' saves at most $n - 1 + (b+2)|C||\mathcal{T}| - |\mathcal{T}| \leq (b+2)|C||\mathcal{T}| - 1 < k$ vertices. ■

Maximum degree $b+2$ and $\deg(s) \leq b+1$. The following theorem shows that the sharp separation between the NP-hardness and polynomiality of Save on trees pointed out in [75] is preserved for any fixed $b \geq 1$.

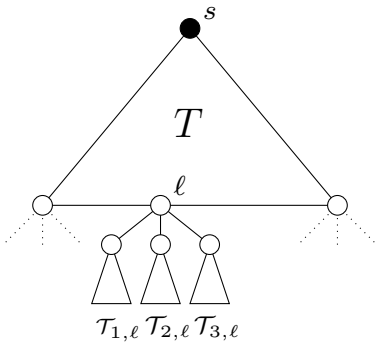


Figure 5.6: Construction of T' from the tree T for the case $b = 1$.

Lemma 62 *Let $b \geq 1$ be any fixed integer and T a tree of maximum degree $b+2$. If the fire breaks out at a vertex of degree at most $b+1$ then all the leaves of T can be saved by a strategy with respect to budget b if and only if T is not complete.*

Proof. Notice that for trees of maximum degree $b+2$, we can protect the vertices in such a way that there is at most one new burning vertex v at each time step. Moreover, the fire stops when the vertex v has degree at most $b+1$.

Suppose that T is not complete. Then there exists a non-leaf vertex v of degree at most $b+1$. From the previous remark we can direct the fire from s to v and stop it. Hence all the leaves of T are saved.

Suppose that T is complete. Then at each time step, there is at least one new burning vertex. Thus there will be a leave which will burn at the end of the process. ■

Since verifying whether a tree is complete can be done in polynomial-time, we deduce the following theorem.

Theorem 63 *For any fixed budget $b \geq 1$,* Save *is polynomial-time solvable for trees of maximum degree $b+2$ if the fire breaks out at a vertex of degree at most $b+1$.*

It is worth noting that Theorem 63 also holds for Firefighter. It suffices to direct the fire to a vertex of degree at most $b+1$ such that the number of burned vertices is minimum.

Theorem 64 *For any fixed budget $b \geq 1$,* Firefighter *is polynomial-time solvable for trees of maximum degree $b+2$ if the fire breaks out at a vertex of degree at most $b+1$.*

5.3.2 Graphs of bounded pathwidth

In this section, we show that the Firefighter problem is NP-complete even on trees of pathwidth three. Then, we provide a polynomial-time algorithm for solving Max Weighted Firefighter on a subclass of trees of pathwidth two, namely k-caterpillars.

Pathwidth ≥ 3. We start with the NP-completeness result. For that purpose we reduce from the NP-hard [88] problem Cubic Monotone 1-In-3-Sat (see Appendix A).

Theorem 65 Firefighter *is* NP*-complete even on trees of pathwidth three and budget one.*

Proof. Clearly, Firefighter belongs to NP. For convenience, we provide a polynomial-time reduction from Cubic Monotone 1-In-3-Sat to Dual Firefighter. This will obviously imply the NP-hardness for Firefighter.

Let ϕ be a formula of Cubic Monotone 1-In-3-Sat with n variables $\{x_1, \ldots, x_n\}$ and m initial clauses $\{c_1, \ldots, c_m\}$. Notice that a simple calculation shows that $n = m$. First, we extend ϕ into a new formula ϕ' by adding m new clauses as follows. For each clause c_j we add the clause $\bar{c}_j$ by taking negation of each variable of c_j. A satisfying assignment for ϕ' is then a truth assignment such that each clause c_j has exactly one true literal and each clause $\bar{c}_j$ has exactly two true literals. It is not hard to see that ϕ has a satisfying assignment if and only if ϕ' has one.

Now we construct an instance $I' = (T, s, 1, k_b)$ of Dual Firefighter from ϕ' as follows (see Figure 5.7). In this construction, adding a guard-vertex g to a vertex v means making a copy of a star with center g and k_b leaves such that g is made adjacent to v. Thus, if we want to burn at most k_b vertices then a guard-vertex needs to be saved or protected. We start with the construction of the tree T.

- Start with a path $\{su_2 = u_1u_2, u_2u_3, \ldots, u_{p-1}u_p\}$ where $p = 2n-1$ and add two degree-one vertices v_{x_i} and $v_{\bar{x}_i}$ adjacent to u_{2i-1} for every $i \in \{1, \ldots, n\}$.

In the following two steps for each $i \in \{1, \ldots, n\}$:

- Add a guard-vertex g_i (resp. $\bar{g}_i$) to v_{x_i} (resp. $v_{\bar{x}_i}$).
- At each vertex v_{x_i} (resp. $v_{\bar{x}_i}$) root a path of length $2 \cdot (n-i)$ at v_{x_i} (resp. $v_{\bar{x}_i}$) in which the endpoint is adjacent to three degree-one vertices (called literal-vertices) denoted by $\ell_1^{x_i}$, $\ell_2^{x_i}$, and $\ell_3^{x_i}$ (resp. $\ell_1^{\bar{x}_i}$, $\ell_2^{\bar{x}_i}$, and $\ell_3^{\bar{x}_i}$). Each literal-vertex corresponds to an occurence of the variable x_i in an initial clause of ϕ'. Analogously, the literal-vertices $\ell_1^{\bar{x}_i}$, $\ell_2^{\bar{x}_i}$, and $\ell_3^{\bar{x}_i}$ represent the negative literal $\bar{x}_i$ that appears in the new clauses.

Notice that each leaf of the constructed tree so far is at distance exactly $p+1$ from s.

- For each variable x_i, $i \in \{1, \ldots, n\}$, choose a clause c_j, $j \in \{1, \ldots, m\}$ containing x_i (resp. $\bar{x}_i$). Then root a path $Q_j^{x_i}$ (resp. $\bar{Q}_j^{x_i}$) of length $j-1$ at $\ell_1^{x_i}$ (resp. $\ell_1^{\bar{x}_i}$), and add a guard-vertex $g_j^{x_i}$ to the endpoint of $Q_j^{x_i}$ (resp. add two degree-one vertices, named dummy-vertices, adjacent to the endpoint of $\bar{Q}_j^{x_i}$). Repeat the same for two other clauses with x_i (resp. $\bar{x}_i$) and root a path at $\ell_2^{x_i}$, $\ell_3^{x_i}$ (resp. $\ell_2^{\bar{x}_i}$, $\ell_3^{\bar{x}_i}$).

To finish the construction, set $k_b = p + \sum_{i=1}^{n}(3 + 2(n-i) + 1) + 3\sum_{i=1}^{m-1} i + 4m$.

In what follows, we use Lemma 58 and thus we only consider strategies that protect a vertex adjacent to a burned vertex at each time step. Recall that the budget is set to one in the instance I'. Now we show that there is a satisfying assignment for ϕ' if and only if there exists a strategy for I' such that at most k_b vertices in T are burned.

"$\Rightarrow$" : Suppose that there is a satisfying assignment τ for ϕ'. We define the following strategy Φ_τ from τ. At each step t from 1 to $p+1$, if t is odd then protect $v_{\bar{x}_{\lceil t/2 \rceil}}$ if $x_{\lceil t/2 \rceil}$ is true otherwise protect $v_{x_{\lceil t/2 \rceil}}$. If t is even then protect the guard-vertex $g_{\lceil t/2 \rceil}$ if $v_{\bar{x}_{\lceil t/2 \rceil}}$ has been protected, otherwise protect $\bar{g}_{\lceil t/2 \rceil}$. At time step $p+1$, the number

of burned vertices is exactly $p+\sum_{i=1}^{n}(3+2(n-i)+1)$. Moreover, the literal-vertices that are burned in T corresponds to the true literals in ϕ'. Thus, by construction and since τ statisfies ϕ', the vertices adjacent to a burning vertex are exactly one guard-vertex, four dummy vertices and $3(n-1)$ other vertices. Since we must protect the guard vertex, we know that $3(n-1)+4$ vertices are going to burn at the next time step. More generally, at any subsequent time step t from $p+1$ to $p+m$ the stategy Φ_τ must protect a guard-vertex $g_{t-p}^{x_i}$ for some x_i, and four dummy-vertices together with $3(n-(t-p))$ vertices get burned. It follows that the number of burned vertices is $\sum_{t=p+1}^{p+m}[3(n-(t-p))+4] = \sum_{t=1}^{m}[3(n-t)+4] = 3\sum_{t=1}^{m-1} t + 4m$ leading to a total of k_b burned vertices.

"$\Leftarrow$": Conversely, assume that there exists a strategy Φ for I' such that at most k_b vertices in T are burned. Observe first that this strategy protects either v_{x_i} or $v_{\bar{x}_i}$ for each $i \in \{1,\ldots,n\}$. As a contradiction, suppose that there exists an $i \in \{1,\ldots,n\}$ such that Φ does not protect v_{x_i} and $v_{\bar{x}_i}$. Then in some time step both v_{x_i} and $v_{\bar{x}_i}$ get burned. Hence, it is not possible to protect both g_i and $\bar{g}_i$ and at least one will burn implying that more than k_b vertices would burn, a contradiction. Now consider the situation at step $p+1$. As previously, the number of burned vertices so far is exactly $p+\sum_{i=1}^{n}(3+2(n-i)+1)$. Let n_g and n_d be the number of guard-vertices and dummy-vertices adjacent to a burned vertex, respectively. Following the discussion, we know that $n_g = 3 - n_d/2$ with $1 \leq n_g \leq 3$ and $1 \leq n_d \leq 6$. In what follows, we will show that exactly one guard-vertex is adjacent to a burned vertex that is $n_g = 1$. First, we must have $n_g \leq 1$ since otherwise more than k_b vertices would burn. Suppose now that $n_g = 0$. Hence we have exactly $n_d = 6 - 2n_g = 6$ dummy vertices adjacent to burned vertices. Therefore the number of burned vertices would be at least $3\sum_{t=1}^{m-1} t + 4(m-1) + 6 = 3\sum_{t=1}^{m-1} t + 4m + 2$ giving us a total of at least $k_b + 2$ burned vertices, a contradiction. Hence, we have $n_g = 1$ and $n_d = 4$. This implies that exactly the three literal-vertices $\ell_1^{x_{i_1}}$, $\ell_1^{\bar{x}_{i_2}}$, $\ell_1^{\bar{x}_{i_3}}$ (resp. $\ell_1^{\bar{x}_{i_1}}$, $\ell_1^{x_{i_2}}$,$\ell_1^{x_{i_3}}$) are burned (resp. saved) where x_{i_1}, x_{i_2}, x_{i_3} belong to the clause c_1. Thus if we set x_{i_1} to true, x_{i_2} to false and x_{i_3} to false in ϕ', we have exactly one true literal in c_1 and exactly two true literals in $\bar{c}_1$. Applying this argument iteratively from step $p+2$ to $p+m$, we arrive at the following satisfying assignment for ϕ': if the vertex v_{x_i} is protected then set x_i to false, otherwise set it to true.

It remains to prove that the pathwidth of T is at most three. To see this, observe that any subtree rooted at x_i or $\bar{x}_i$ has pathwidth two. Let P_{x_i} and $P_{\bar{x}_i}$ be the paths of the path-decompositions of these subtrees, respectively. We construct the path-decomposition for T as follows. For every $i \in \{1,\ldots,n-1\}$, define the node $B_i = \{u_{2i-1}, u_{2i}, u_{2i+1}\}$. Extend all nodes of the paths P_{x_i} and $P_{\bar{x}_i}$ to P'_{x_i} and $P'_{\bar{x}_i}$ by adding the vertex u_{2i-1} inside it. Finally, connect the paths P'_{x_1}, $P'_{\bar{x}_1}$ and the node B_1 to form a path and continue in this way with P'_{x_2}, $P'_{\bar{x}_2}$, B_2, P'_{x_3}, $P'_{\bar{x}_3}$, B_3, …, B_{n-1}, P'_{x_n}, $P'_{\bar{x}_n}$. This completes the proof. ■

The above proof can also be regarded as a simpler proof of the NP-completeness of the Firefighter problem on trees. We can generalize the previous result to any fixed budget $b \geq 1$ as follows.

Corollary 66 *For any fixed budget $b \geq 1$,* Firefighter *is* NP*-complete even on trees of pathwidth three.*

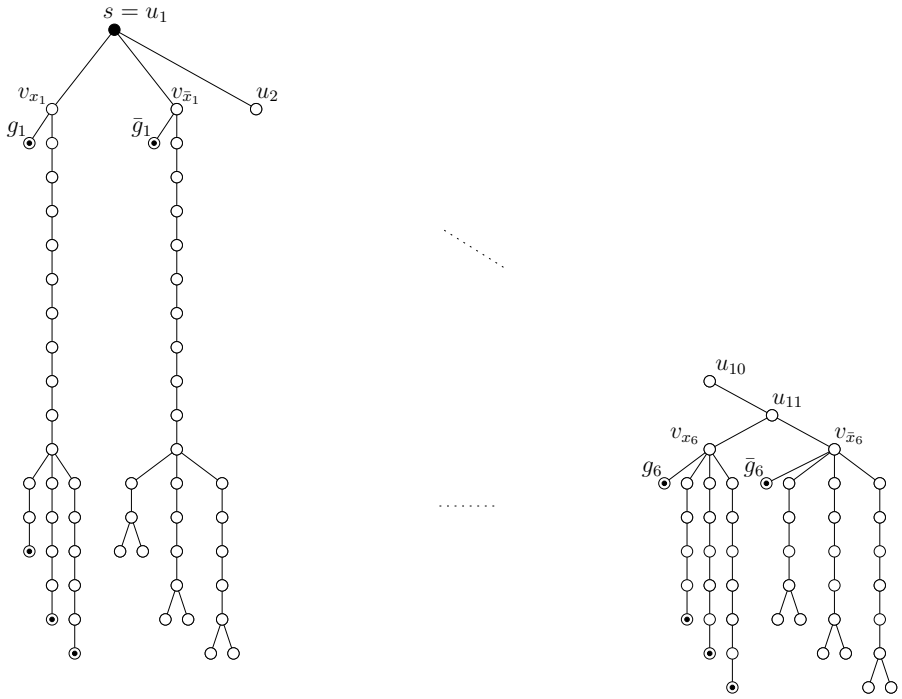


Figure 5.7: An example of part of a tree constructed from the formula $\phi = (x_3 \vee x_4 \vee x_5) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (x_1 \vee x_3 \vee x_6) \wedge (x_1 \vee x_4 \vee x_6) \wedge (x_2 \vee x_6 \vee x_5)$. Guard vertices are represented by a dot within a circle.

Proof. We start from the reduction of Theorem 65 and alter the tree T as follows. Add a path $\{sw_2 = w_1w_2, w_2w_3, \ldots, w_{2n-1}w_{2n}\}$ to T together with $b-1$ guard-vertices added to each w_i. First, one can easily check that the pathwidth remains unchanged since the added component has pathwidth two and is only connected to the root s. Second, it can be seen that at each time step, only one firefighter can be placed "freely" as the other $b-1$ firefighters must protect $b-1$ guard-vertices. It follows that we end up to a similar proof as for Theorem 65. This completes the proof. ■

k-caterpillars. We now present a subclass of trees of pathwidth two for which Max Weighted Firefighter is polynomial-time solvable. We first prove the result for Max 0-1 Firefighter and further extend it to the general weighted version.

MacGillivray and Wang [82] showed that the degree greedy algorithm (at each time step, protect the vertex adjacent to a burned vertex with the highest degree) gives an optimal solution for Max Firefighter on caterpillars when the budget is $b = 1$. However, this result does not hold anymore for k-caterpillars as can be seen in Figure 5.8.

In order to prove our main result of this section we first need to state the following result.

Theorem 67 *For any $k \geq 1$ and any budget $b \geq 1$,* Max 0-1 Firefighter *is polynomial-time solvable for k-stars.*

Proof. We construct a polynomial-time reduction from Max 0-1 Firefighter to the Min Cost Flow problem (see Appendix A) which is known to be polynomial-time solvable [5]. Let $I = (G = (V, E), s, w, b)$ be an instance of Max 0-1 Firefighter

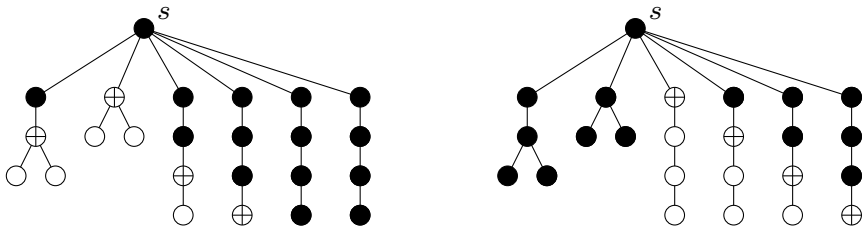


Figure 5.8: A k-caterpillar for which the degree greedy algorithm (left) saves less vertices than the optimal solution (right).

where G is a k-star. First assume that $s \in V$ is the center of G. Let $d = \deg_G(s)$ and S be the set of vertices with weight equal to one. Let $P_1 = \{sv_{11}, v_{11}v_{21}, \ldots, v_{(p_1-1)1}v_{p_11}\}$, $\ldots$, $P_d = \{sv_{1d}, v_{1d}v_{2d}, \ldots, v_{(p_d-1)d}v_{p_dd}\}$ be the maximal paths of G starting at vertex s, with $p_1, \ldots, p_d \leq k$ and $v_{0j} = s$, for $j = 1, \ldots, d$, if it exists. Let $p = \max\{p_1, \ldots, p_d\}$. For each vertex v_{ij} in these paths, we define $S_{ij} = \{v_{ij}, v_{(i+1)j}, \ldots v_{p_jj}\} \cap S$. Notice that we may assume that every path P_j contains at least one vertex of S (otherwise we may delete $V(P_j) \setminus \{s\}$). We construct the instance $I' = (G' = (V', A), \alpha, \beta, \gamma)$ of Min Cost Flow as follows.

- Construct the digraph $G' = (V', A)$ (see Figure 5.9), where $V' = \{L_1, \ldots, L_p\} \cup \{C_1, \ldots, C_d\} \cup \{\ell, r\}$ and $A = \{(L_i, C_j) : v_{ij} \in P_j\} \cup \{(\ell, L_i) : i = 1, \ldots, p\} \cup \{(C_j, r) : j = 1, \ldots, d\}$.
- Associate with each arc (L_i, C_j) of G', a cost $\beta(L_i, C_j) = -|S_{ij}|$. All other arcs have cost zero.
- Associate with each arc (ℓ, L_i) a capacity $\alpha(\ell, L_i) = b$, with each arc (L_i, C_j) a capacity $\alpha(L_i, C_j) = 1$ and with each arc (C_j, r) a capacity $\alpha(C_j, r) = 1$.
- Associate a supply of value d with vertex ℓ and a demand of value $-d$ with vertex r (all other vertices have a supply and a demand equal to zero).

This completes the construction and clearly I' can be obtained from I in polynomial time. Now we claim that solving the instance I of Max 0-1 Firefighter is equivalent to solving the instance I' of Min Cost Flow.

"$\Rightarrow$": Consider a feasible solution of Max 0-1 Firefighter in I of value ν. We may assume without loss of generality (see Lemma 58) that at most one vertex is protected in each path P_j, $j \in \{1, \ldots, d\}$, and (see Lemma 58) that at most b vertices are protected in each set $V_i = \{v_{i1}, v_{i,2}, \ldots, v_{id}\}$, $i \in \{1, \ldots, p\}$ (notice that some of these vertices $v_{ij}, j = 1, \ldots, d$, may not exist in G). Let $\mathcal{D} = \{v_{ij} : v_{ij} \text{ is protected}, i \in \{1, \ldots, p\}, j \in \{1, \ldots, d\}\}$. Thus $\nu = \sum_{v_{ij} \in \mathcal{D}} |S_{ij}|$. Consider now some vertex $v_{ij} \in \mathcal{D}$. Then in G', we will use one flow unit on the path ℓ-L_i-C_j-r. Repeating this procedure for every vertex in $\mathcal{D}$, we obtain a flow in G' of value $|\mathcal{D}|$ and of cost $\sum_{v_{ij} \in \mathcal{D}} \beta(L_i, C_j) = \sum_{v_{ij} \in \mathcal{D}} -|S_{ij}| = -\nu$. Since at most b vertices are protected in each set V_i, it follows that at most b units of flow use the arc (ℓ, L_i), for $i \in \{1, \ldots, p\}$. Furthermore, since exactly one vertex is protected in each path P_j, it follows that exactly one flow unit uses the arc (C_j, r) for $j \in \{1, \ldots, d\}$. Hence, we obtain a feasible solution of Min Cost Flow in I' of value $-\nu$.

"$\Leftarrow$": Conversely, consider now a feasible solution of Min Cost Flow in I' of value $-\mu$. Let $\mathcal{A}$ be the set of arcs (L_i, C_j) used by a flow unit, $i \in \{1, \ldots, p\}$,

$j \in \{1, \ldots, d\}$. Thus $-\mu = \sum_{(L_i, C_j) \in \mathcal{A}} -|S_{ij}|$. For each flow unit on a path ℓ-L_i-C_j-r, we choose vertex v_{ij} in G to be protected, for $i \in \{1, \ldots, p\}$, $j \in \{1, \ldots, d\}$. Since the capacity of an arc (ℓ, L_i) is b, at most b vertices in V_i will be chosen to be protected, $i \in \{1, \ldots, p\}$. Let us denote by V_i^* the set of vertices in V_i chosen to be protected. Furthermore, since the capacity of an arc (C_j, r) is one, exactly one vertex in each path P_j will be chosen to be protected, $j \in \{1, \ldots, d\}$. Thus, if we protect at each time step i the vertices in V_i^*, we obtain a feasible solution of Max 0-1 Firefighter in I of value $\sum_i \sum_{v_{ij} \in V_i^*} |S_{ij}| = \mu$.

Finally, we have to consider the case when s is not the center of G. The case when s has degree one is trivial. Thus we may assume now that $\deg_G(s) = 2$. If $b \geq 2$, we are done. Thus we may assume now that $b = 1$. If both neighbors of s are in S, then the optimal solution value is clearly $|S| - 1$. If both neighbors of s are not in S, then the optimal solution value is clearly $|S|$. Hence the only case remaining is when exactly one neighbor of s is in S. Let u_1, u_2 be the neighbors of s such that $u_1 \in S, u_2 \notin S$. If u_2 is not the center of G, the optimal solution value is clearly $|S|$. Thus we may assume now that u_2 is the center of G. Let Q denote the set of vertices of the unique maximal path starting at vertex u_2 and containg u_1. In that case we have to compare the value of two solutions: (i) $|S| - 1$ which is the value of the solution obtained by protecting first u_2 and then, during the second time step, we protect the neighbor of u_1 which is not s (if it exists); (ii) the value of the solution obtained by protecting first u_1 and then applying our algorithm described above to the graph $G - (Q \setminus \{u_2\})$ (*i.e.,* by reducing our problem to a Min Cost Flow problem). ■

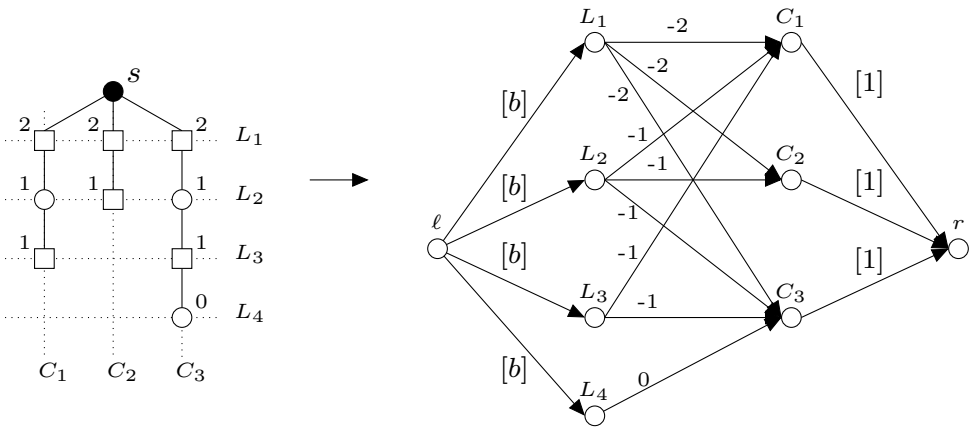


Figure 5.9: The digraph G' (right) obtained from G (left). For the graph G', the capacities are indicated within brackets while the other numbers correspond to the costs. In the graph G, the squares correspond to vertices with weight one. Furthermore, the value associated to each vertex is equal to the number of saved vertices if this vertex is protected.

Notice that the polynomial reduction from Max 0-1 Firefighter to Min Cost Flow described in the proof of Theorem 67 is still valid if the number of vertices that can be protected at each time step is not constant (for instance if we are allowed to protect at most b_1 vertices during the first time step, b_2 vertices during the second time step, etc.). In that case we just need to adapt the capacity of the arcs (ℓ, L_i) accordingly.

Furthermore the polynomial reduction remains valid in the case where some of the vertices in a set V_i are not allowed to be protected during time step i. In this case we

simply do not put an arc from L_i to the corresponding vertices C_j in G'.

Consider now a k-caterpillar $G = (V, E)$. Let P be the path in the caterpillar from which G has been obtained, which is induced by vertices of degree at least two. We will call P the *spine* of the k-caterpillar.

We are now ready to prove the main result of this section.

Theorem 68 *For any $k \geq 1$ and any budget $b \geq 1$,* Max 0-1 Firefighter *is polynomial-time solvable for k-caterpillars.*

Proof. Let $G = (V, E)$ be a k-caterpillar and let $P = \{v_1v_2, v_2v_3, \ldots, v_{p-1}v_p\}$ be the spine of G. First assume that s is a vertex of P, say $s = v_i$, $i \in \{1, \ldots, p\}$. Let $P_1 = \{v_1v_2, \ldots, v_{i-2}v_{i-1}\}$ and $P_2 = \{v_{i+1}v_{i+2}, \ldots, v_{p-1}v_p\}$. It follows from Lemma 58 that we may assume that at most one vertex is protected in P_1 and at most one vertex is protected in P_2. Consider a strategy in which we decide to protect exactly two vertices of P, say vertex v_j, for $j \in \{1, \ldots, i-1\}$ and vertex v_q, for $q \in \{i+1, \ldots, p\}$. We may assume that v_j is protected during time step $i - j$ and vertex v_q is protected during time step $q - i$ (see Lemma 58). Notice that the vertices $v_{j+1}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{q-1}$ will not be protected in this strategy. Construct a $(k + p)$-star G' as follows (see Figure 5.10):

- Delete all vertices $v_1, \ldots, v_j$ as well as the legs at these vertices (all these vertices are saved in our strategy).
- Delete all vertices $v_q, \ldots, v_p$ as well as the legs at these vertices (all these vertices are saved in our strategy).
- Delete all edges of P.
- For every $r \in \{j+1, \ldots, i-1, i+1, \ldots, q-1\}$, let $u^r_1, \ldots, u^r_{\deg_G(v_r)-2}$ be the neighbors of v_r not belonging to P; delete v_r and replace it by $\deg_G(v_r) - 2$ vertices $v^r_1, \ldots, v^r_{\deg_G(v_r)-2}$ such that v^r_l is adjacent to u^r_l for $l \in \{1, \ldots, \deg_G(v_r) - 2\}$.
- Join every vertex v^r_ℓ, for $r \in \{j + 1, \ldots, i - 1\}$ and $\ell \in \{1, \ldots, \deg_G(v_r) - 2\}$, to v_i by a path $P^{r\ell}$ of length $i - r$.
- Join every vertex v^r_ℓ, for $r \in \{i + 1, \ldots, q - 1\}$ and $\ell \in \{1, \ldots, \deg_G(v_r) - 2\}$, to v_i by a path $P^{r\ell}$ of length $r - i$.

From the above construction it follows that G' is a $(k + p)$-star with center v_i. Now in order to solve our initial problem, we need to solve Max 0-1 Firefighter in G' with the following additional constraints: for every $r \in \{j + 1, \ldots, i - 1, i + 1, \ldots, q - 1\}$ and every $\ell \in \{1, \ldots, \deg_G(v_r) - 2\}$ we are not allowed to protect the vertices of $V(P^{r\ell})$. Indeed, since we decided to protect v_j and v_q, the vertices $v_{j+1}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{q-1}$ will not be saved. Notice that these vertices are represented by the vertices of paths $P^{r\ell}$ in G'. Moreover, if $i - j \neq q - j$ then at time steps $i - j$ and $q - j$ only $b - 1$ firefighters are available (since we protect v_j and v_q at these time steps); if $i - j = q - j$ then only $b - 2$ firefighters are available at time step $i - j$. It follows from Theorem 67 and previous discussion that this problem can be solved in polynomial time.

Since the number of choices of a pair of vertices (v_j, v_q) to be protected on P is $(i - 1) \times (p - i)$, we can determine in polynomial time the best strategy to adopt if we want to protect exactly two vertices on P. Notice that a similar procedure to the one described above can be used if we decide to protect exactly one vertex on

P respectively if we decide not to protect any vertex of P. Clearly the number of choices of exactly one vertex v_j, $j \in \{1, \ldots, i-1, i+1, \ldots, p\}$, to be protected on P is $p-1$. Thus we conclude that if $s \in V(P)$ we can determine an optimal strategy in polynomial time.

It remains the case when $s \notin V(P)$. Similar to the proof of Theorem 67, we will distinguish several cases. The case when s has degree one is trivial. Thus we may assume now that $\deg_G(s) = 2$. If $b \geq 2$, we are done. Thus we may assume now that $b = 1$. Let S be the set of vertices with weight equal to one. If both neighbors of s are in S, then the optimal solution value is clearly $|S| - 1$. If both neighbors of s are not in S, then the optimal solution value is clearly $|S|$. Hence the only case remaining is when exactly one neighbor of s is in S. Let u_1, u_2 be the neighbors of s such that $u_1 \in S, u_2 \notin S$. If $u_2 \notin V(P)$, the optimal solution value is clearly $|S|$. Thus we may assume now that $u_2 \in V(P)$. In this case we have to compare the value of two solutions: (i) $|S| - 1$ which is the value of the solution obtained by protecting first u_2 and then, during the second time step, we protect the neighbor of u_1 which is not s (if it exists); (ii) the value of the solution obtained by protecting first u_1 and then applying our algorithm described above to the graph $G - (Q \setminus \{u_2\})$, where Q is the set of vertices of the unique maximal path starting at u_2 and containing u_1. ■

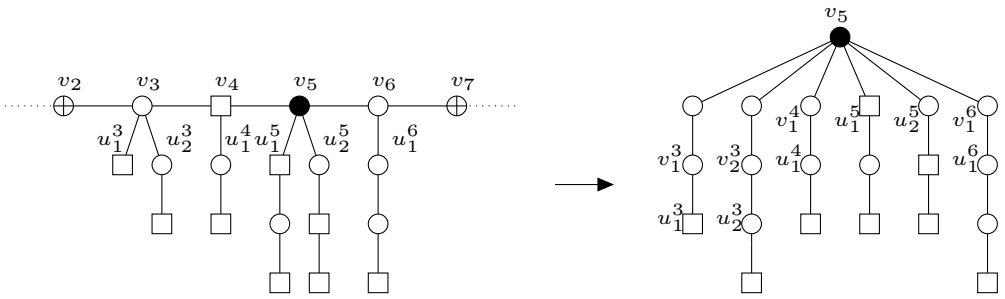


Figure 5.10: The construction of G' with $i = 5$, $j = 2$, and $q = 7$. The squares correspond to vertices with weight one.

Extension to general weights. We would like to mention that some of our positive results (Theorem 64, Theorem 67, and Theorem 68) may be generalized to a weighted version.

Using the strategy in the proof of Theorem 64, if we direct the fire to a vertex of degree at most $b+1$ such that the total weight of the burned vertices is minimum then we get the following result.

Theorem 69 *For any budget $b \geq 1$,* Max Weighted Firefighter *is polynomial-time solvable for trees of maximum degree $b+2$ if the fire breaks out at a vertex of degree at most $b+1$.*

Now by replacing the costs $\beta(L_i, C_j)$ in the proof of Theorem 67 by $\beta(L_i, C_j) = -|\sum_{v \in S_{ij}} w(v)|$ and adapting the case when s is not the center of G according to the weights, it is not difficult to see that we obtain the following.

Theorem 70 *For any $k \geq 1$ and any budget $b \geq 1$,* Max Weighted Firefighter *is polynomial-time solvable for k-stars.*

Using this result and adapting the case when $s \notin V(P)$ according to the weights, it is straightforward that we obtain the following result.

Theorem 71 *For any $k \geq 1$ and any budget $b \geq 1$,* Max Weighted Firefighter *is polynomial-time solvable for k-caterpillars.*

5.3.3 Path-like graphs of bounded degree

As previously shown, for any fixed budget $b \geq 1$, Firefighter is NP-complete on trees of bounded degree $b+3$ (Theorem 61) and on trees of bounded pathwidth three (Theorem 65). It is thus natural to ask the complexity of the problem when both the degree and the pathwidth of the input graph are bounded. In what follows, we answer this question positively.

Theorem 72 *Consider a graph of pathwidth* pw *and maximum degree Δ. If the number of initially burned vertices is bounded by $g_1(\text{pw}, \Delta)$ for some function g_1 then there exists a protection strategy where at most $g_2(\text{pw}, \Delta)$ vertices get burned for some function g_2.*

Proof. We first prove the following claim: Consider a graph of cutwidth cw. If the number of initially burned vertices is bounded by $g_1(\text{cw})$ for some function g_1 then there exists a protection strategy where at most $g_2(\text{cw})$ vertices get burned for some function g_2. We will prove this by induction on cw.

The claim is obviously true when $\text{cw} = 0$ since the graph cannot contain any edge. Suppose now that the property is true for any graph of cutwidth at most k, $k > 0$. We show that it also holds for a graph of cutwidth $k+1$. Let $H = (V, E)$ be such a graph, and $F \subseteq V$ be the set of initially burned vertices with $|F| \leq g_1(\text{cw}(H))$ for some function g_1. Consider a linear layout $L = (v_1, \ldots, v_n)$ of H such that for every $i = 1, \ldots, n-1$, there are at most $k+1$ edges with one endpoint in $\{v_1, \ldots, v_i\}$ and the other in $\{v_{i+1}, \ldots, v_n\}$. For every $s \in F$, we define the following sets.

$$R_i(s) = \begin{cases} \{s = v_k, v_{k+1}, \ldots, v_{k'}\} & \text{if } \exists v_{k'} \in N^i[s] : v_{k'} = \arg\max_{v \in N^i[s]} d_L(s, v) \\ R_{i-1}(s) & \text{otherwise} \end{cases} \tag{5.1}$$

$$L_i(s) = \begin{cases} \{s = v_k, v_{k-1}, \ldots, v_{k'}\} & \text{if } \exists v_{k'} \in N^i[s] : v_{k'} = \arg\min_{v \in N^i[s]} d_L(s, v) \\ L_{i-1}(s) & \text{otherwise} \end{cases} \tag{5.2}$$

We are now in position to define the set $B_i(s)$, called a *bubble*, by $B_i(s) = L_i(s) \cup R_i(s)$. Informally speaking, a bubble $B_i(s)$ corresponds to the *effect zone* of s after i steps of propagation *i.e.* every burned vertex inside the bubble is due to the vertex s. The idea of the proof is then to show that every bubble can be "isolated" from the rest of the graph in a bounded number of steps by surrounding it with firefighters (see Figure 5.11). We then show that the inductive hypothesis can be applied on each bubble which will prove the theorem.

Let $s_1, s_2 \in F$. We say that two bubbles $B_i(s_1)$ and $B_j(s_2)$ for some $i, j \geq 0$ *overlap* if there exists an edge $uv \in E$ with $u \in B_i(s_1)$ and $v \in B_j(s_2)$. In this case, we can *merge* the two bubbles into one *i.e.* we create a new bubble which is the union of $B_i(s_1)$ and $B_j(s_2)$.

Let us consider an initially burned vertex $s \in F$ and its bubble $B_{2\cdot\mathrm{cw}(H)}(s)$. First, merge $B_{2\cdot\mathrm{cw}(H)}(s)$ with every other bubble $B_{2\cdot\mathrm{cw}_H}(s')$ with $s' \in F$ that possibly overlap into a new one $B'_{2\cdot\mathrm{cw}(H)}(s)$. By definition, we know that the number of edges with an endpoint in $B'_{2\cdot\mathrm{cw}(H)}(s)$ and the other one in $V \setminus B'_{2\cdot\mathrm{cw}(H)}(s)$ is less or equal to $2\cdot\mathrm{cw}(H)$. Thus, we define the strategy that consists in protecting one vertex $v \in V \setminus B'_{2\cdot\mathrm{cw}(H)}(s)$ adjacent to a vertex in $B'_{2\cdot\mathrm{cw}(H)}(s)$ at each step $t = 1, \ldots, 2\cdot\mathrm{cw}(H)$. Let F' be the set of vertices burned at step $2\cdot\mathrm{cw}(H)$. Since $\Delta(H) \leq 2\cdot\mathrm{cw}(H)$, we deduce that $|F'|$ is less or equal to $|F| \cdot \Delta_H^{2\cdot\mathrm{cw}(H)} \leq g_1(\mathrm{cw}(H)) \cdot (2\cdot\mathrm{cw}(H))^{2\cdot\mathrm{cw}(H)} = h(\mathrm{cw}(H))$ for some function h. Let us consider the subgraph $H' = H[B'_{2\cdot\mathrm{cw}(H)}(s)]$. Observe that we can safely remove every edge uv from H' for which $u, v \in F'$. Indeed, such edge cannot have any influence during the subsequent steps of propagation. By the definition of a bubble, this implies that the cutwidth of H' is decreased by one and thus is now at most k. Therefore, we can apply our inductive hypothesis to H' which tells us that there is a strategy for H' such that at most $g'_2(\mathrm{cw}_{H'})$ vertices are burned for some function g'_2. By Lemma 56, this strategy uses at most $g'_2(\mathrm{cw}_{H'})$ steps to be applied. It follows that the number of burned vertices in H after applying this strategy is at most the number of burned vertices from step 1 to $2\cdot\mathrm{cw}(H) + g'_2(\mathrm{cw}(H'))$ which is $|F|\cdot\Delta(H)^{2\cdot\mathrm{cw}(H)+g'_2(\mathrm{cw}_{H'})} \leq g_1(\mathrm{cw}(H))\cdot(2\cdot\mathrm{cw}(H))^{2\cdot\mathrm{cw}(H)+g'_2(\mathrm{cw}(H'))} = h(\mathrm{cw}(H))$ for some function h. From now on, one can see that the previous argument can be applied iteratively to each bubble. Since the number of bubbles is bounded by $g_1(\mathrm{cw}(H))$ (there is at most one bubble for each vertex initially on fire), we deduce that the total number of burned vertices is bounded by some function of $\mathrm{cw}(H)$. This concludes the proof of the claim.

We are now in position to show the theorem. Let G be a graph. Suppose that the number of initially burned vertices in G is at most $g_1(\mathrm{pw}(G), \Delta(G))$ for some function g_1. We know that $\mathrm{pw}(G) \leq \mathrm{cw}(G)$ and $\Delta(G) \leq 2\cdot\mathrm{cw}(G)$ [78]. Thus the number of burned vertices is at most $g'_1(\mathrm{cw}(G))$ for some function g'_1. From the above claim, we deduce that there exists a strategy such that at most $g'_2(\mathrm{cw}(G))$ vertices get burned. Since $\mathrm{cw}(G) \leq \mathrm{pw}(G)\cdot\Delta(G)$ [42], it follows that the number of burned vertices is bounded by $g_2(\mathrm{pw}(G), \Delta(G))$ for some function g_2. This completes the proof. ■

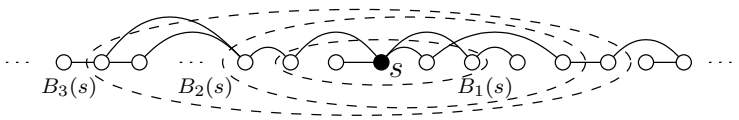


Figure 5.11: A linear layout of a graph of cutwidth two. Dashed ellipses represent the bubbles associated to an initially burned vertex s.

We choosed to present the following parameterized complexity results here rather than in the later dedicated section. The reason is that these results complete quite well the complexity analysis conducted so far. In the next section (Section 5.4), we show that Dual Firefighter is fixed-parameter tractable with respect to the combined parameter k_b and "maximum degree" of the input graph. Thus, using Theorem 72 together with Corollary 85, we deduce the following.

Theorem 73 FIREFIGHTER *is fixed-parameter tractable with respect to the combined parameter "pathwidth" and "maximum degree" of the input graph.*

From the claim in Theorem 73, we easily deduce the following.

Corollary 74 FIREFIGHTER *is fixed-parameter tractable with respect to the parameter "cutwidth" of the input graph.*

Finally, since for any graph G it holds that $\mathrm{cw}(G) \leq \frac{\mathrm{bw}(G)(\mathrm{bw}(G)+1)}{2}$ [18], we have the following result.

Corollary 75 FIREFIGHTER *is fixed-parameter tractable with respect to the parameter "bandwidth" of the input graph.*

5.4 Parameterized complexity in the general case

In this section, we give upper and lower bounds on parameterized complexity of the following three problems: FIREFIGHTER parameterized by k, BOUNDED FIREFIGHTER parameterized by k_p, and DUAL FIREFIGHTER parameterized by k_b.

5.4.1 Firefighter

In the following, we will give the parameterized complexity upper and lower bounds of FIREFIGHTER with respect to its standard parameterization.

Theorem 76 FIREFIGHTER *is solvable in* $n^{O(k)}$ *time.*

Proof. Let (G, s, b, k) be an instance of FIREFIGHTER. We run the $n^{O(k_p)}$-time algorithm of Theorem 79 for all $k_p = 1, \ldots, k$. Observe that it is possible to save k vertices of the graph if and only if the algorithm saves at least k vertices for some value of k_p. This implies a running time of $n^{O(k)}$. ■

Next, we prove in the following that the above algorithm is nearly optimal. Recall that ETH (*Exponential-Time Hypothesis*) is an assumption stating that the 3-SAT problem cannot be solved in subexponential time in the worst case [68].

Theorem 77 *For any fixed budget* $b \geq 1$, FIREFIGHTER *is* W[1]*-hard with respect to the parameter* k *and cannot be solved in* $n^{o(\sqrt{k})}$ *time even on bipartite graphs unless* ETH *fails.*

Proof. We construct an fpt-reduction from CLIQUE to FIREFIGHTER as follows. Let $(G = (V, E), k)$ be an instance of CLIQUE. We construct the instance (G', s, b, k') of FIREFIGHTER as follows. (see Figure 5.12).

- For each edge $uv \in E$, we add a vertex e_{uv}; this set of vertices is denoted by F.
- Add b copies $V^1, \ldots, V^b$ of V, *i.e.* , for each vertex $v \in V$, we add vertices $v^1 \in V^1, \ldots, v^b \in V^b$.
- Add an edge from e_{uv} to both u^h and v^h for each $uv \in E$ and each $h = 1, \ldots, b$.
- Add a vertex s, and add vertices $a_{i,j}$ for all $1 \leq i \leq k-1$ and $1 \leq j \leq (k-1)b+1$.

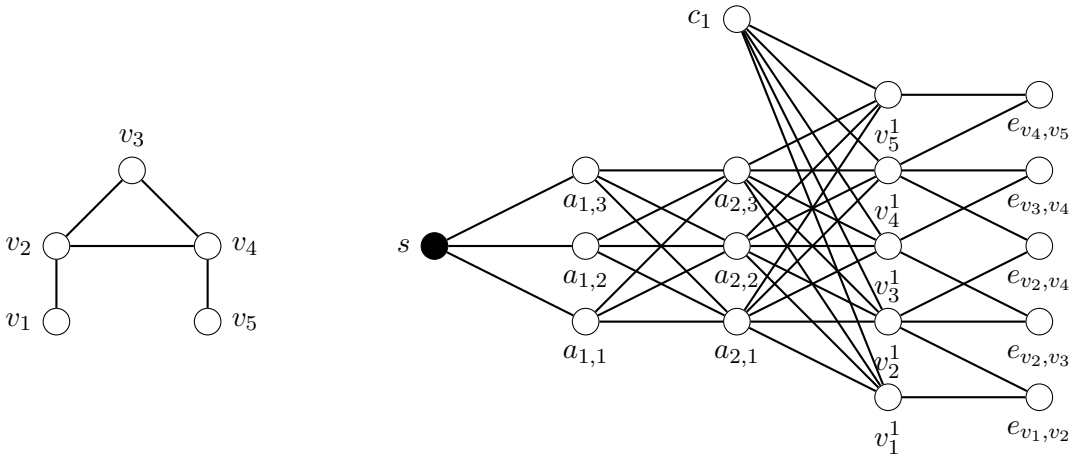


Figure 5.12: An instance of Clique and the corresponding graph G' constructed in the proof of Theorem 77 for $k = 3$ and $b = 1$.

- Connect $a_{i,j}$ to $a_{i',j'}$ $(i' = i + 1)$ for all i, j, j', connect $a_{1,j}$ to s for all j, and connect $a_{k-1,j}$ to each vertex of $V' = \bigcup_{1 \leq h \leq b} V^h$ for all j.

- Add b vertices $c_1, \ldots, c_b$ adjacent to all vertices of V'.

To finish the construction, set $k' = kb + \binom{k}{2} + b$.

We claim that (G, k) is a yes-instance if and only if (G', s, b, k') is a yes-instance.

"⇒": Suppose that G has a clique $C \subseteq V$ of size k. Then the strategy that protects the vertices $v^1, \ldots, v^b$ for all $v \in C$ saves the vertices e_{uv} for all $u, v \in C$. Since C is a clique, these vertices e_{uv} are indeed present in G'. Additionally, we can protect (and thus save) vertices $c_1, \ldots, c_b$. It follows that this strategy saves at least k' vertices.

"⇐": Conversely, suppose that there exists a strategy $\Phi = \{D_1, \ldots, D_n\}$ for (G', s, k', b) that saves at least k' vertices where $D_i = \{p_i^1, \ldots, p_i^b\}$ for each $i = 1, \ldots, n$. First observe that if $p_t^h = a_{i,j}$ for some i ,j, and h, then this vertex is not helpful, as there is always a vertex $a_{i,j'}$ that will be burned at time t and has the same neighborhood as $a_{i,j}$. Hence we can assume that no vertex $a_{i,j}$ is protected by the strategy. This implies that all vertices of V' will be burned, except those that are protected by the strategy. But then protecting vertices of F does not save any further vertices. Since the fire will reach V' in k time steps, and thus F in $k + 1$ time steps, the vertices in $(D_1 \cup \ldots \cup D_n) \cap V'$ are responsible for saving $\binom{k}{2}$ vertices, which is only possible if the vertices of $(D_1 \cup \ldots \cup D_n) \cap V'$ induce a clique of size k in G.

Notice that the parameters in this reductions are quadratically related. Since Clique is W[1]-hard with respect to k and cannot be solved in time $n^{o(k)}$ unless ETH fails [33], we obtain the desired lower bound. ■

The above reduction is also a NP-hardness reduction, and simpler than the original one on bipartite graphs [82].

5.4.2 Bounded Firefighter

We now consider the parameterized complexity of Bounded Firefighter parameterized by k_p. To this end, we introduce the notion of *valid protecting set*. A valid protecting set is a subset of vertices D of a graph G such that there exists a strategy that protects

exactly those vertices in D if the fire starts at some vertex s.

Lemma 78 *Let $G = (V, E)$ be a graph with an initially burned vertex $s \in V$. Verifying whether a subset $D \subseteq (V \setminus \{s\})$ is a valid protecting set can be done in linear time.*

Proof. In this proof, we use the constrained protection phase. Let $L_i = \{v \in V : \text{dist}_{G \setminus \bigcup_{0 \leq j \leq i-1} L_j \cap D}(s, v) = i\}$ for any $i > 0$ and $L_0 = \{s\}$. The graph $G \setminus \bigcup_{0 \leq j \leq i-1} L_j \cap D$ is obtained from G by removing protected vertices from time step 1 through $i-1$. Let $r_i = ib - |\bigcup_{0 \leq j < i} L_j \cap D|$ be the number of available firefighters in step i. If there exists $i \in \{1, \ldots, |V|\}$ such that $|D \cap L_i| > r_i$ or $r_i < 0$, then there is no strategy with respect to budget b that protects D. ■

We are now in position to state the following result.

Theorem 79 Bounded Firefighter *is solvable in $n^{O(k_p)}$ time.*

Proof. Let $(G = (V, E), s, k, b, k_p)$ be an instance of Bounded Firefighter. Among all valid protecting set $D \subseteq (V \setminus \{s\})$ such that $|D| \leq k_p$, the algorithm simply chooses the first one that saves at least k vertices. From Lemma 78, the running time is $n^{O(k_p)}$. ■

By carrying out some modifications in the proof of Theorem 77, we can show that the above algorithm has nearly the best running time.

Theorem 80 *For any fixed budget $b \geq 1$,* Bounded Firefighter *is* W[1]*-hard with respect to the parameter k_p and cannot be solved in $n^{o(\sqrt{k_p})}$ time even on bipartite graphs unless* ETH *fails.*

Proof. Given an instance $I = (G, k)$ of Clique we construct the instance $I' = (G, s, k', b, k_p)$ where G is the same bipartite graph as in the proof of Theorem 77. We set $k_p = kb + b$ and $k' = kb + \binom{k}{2} + b$. Correctness now follows straightforwardly from the arguments in the proof of Theorem 77. ■

5.4.3 Dual Firefighter

In the following subsection, we consider the Dual Firefighter problem. We first show that Dual Firefighter can be solved in $n^{O(k_b)}$ time. Analogously to the previous result, we introduce the notion of *valid burning set*. A valid burning set is a subset B of vertices of a graph G such that there exists a strategy for which, at the end of the process, the burned vertices are exactly those in B if the fire starts at some vertex s.

Lemma 81 *Let $G = (V, E)$ be a graph with an initially burned vertex $s \in V$. Verifying whether a subset $B \subseteq V$ is a valid burning set can be done in linear time.*

Proof. In this proof, we use the constrained protection phase. First of all, if $s \notin B$ or $G[B]$ is not connected then return "no". We now suppose that $s \in B$ and $G[B]$ is connected. Observe that the set of protected vertices must be exactly $N(B)$. For any $v \in N(B)$, let $d_s(v)$ be the length of a shortest path with endpoints s and v in G whose internal vertices are all in B. Then v has to be protected before or at

time step $d_s(v)$. It follows that B is a valid burning set if and only if the number of vertices v for which $d_s(v) \leq t$ is at most bt for every $t = 1, \ldots, k$. Finally, we note that $d_s(v)$ can be easily computed using a breadth-first search. Hence we can determine whether B is a valid burning set in linear time. ■

Theorem 82 Dual Firefighter *is solvable in* $n^{O(k_b)}$ *time.*

Proof. Let $(G = (V, E), s, b, k_b)$ be an instance of Dual Firefighter. Observe that if a vertex of G is burning and is at distance k_b from s then we can deduce that at least $k_b + 1$ are burned. Therefore, we can restrict valid burning sets to those which are subset of the k_b^{th} neighborhood of s. Having said that, exhaustively consider all subsets $B \subseteq N^{k_b}[s]$ with $|B| \leq k_b$. If B is a valid burning set, then the answer is "yes". If no valid burning set is found, then the answer is "no". It follows from Lemma 81 that the running time is $n^{O(k_b)}$. It is worth noting that this time can be rewritten as $\Delta(G)^{k_b^2} \cdot n$ since $|N^{k_b}[s]| \leq \Delta(G)^{k_b}$. ■

We now show that the above algorithm is likely to have the optimal running time.

Theorem 83 Dual Firefighter *is* W[1]*-hard with respect to the parameter* k_b *and cannot be solved in* $n^{o(k_b)}$ *time even on bipartite graphs unless* ETH *fails.*

Proof. We construct an fpt-reduction from the Clique problem (see Appendix A) on regular graphs to Dual Firefighter. Let (G, k) be an instance of Clique where G is a Δ-regular graph. We construct the instance (G', s, b, k_b) of Dual Firefighter from (G, k) as follows. Add a new vertex s adjacent to all vertices of G. Set $k_b = k+1$ and $b = b_1 + b_2$ where $b_1 = k(n-k)$ and $b_2 = k\Delta - \binom{k}{2}$. In what follows, attaching a guard vertex g to a vertex v means making a copy of a star with center g and $b + k_b$ leaves such that g is made adjacent to v. Thus, if we want to burn at most k_b vertices then a guard vertex needs to be saved or protected. Now, attach $b - (n-k)$ guard vertices to s and $n-k$ guard vertices to every vertex in V as well. Remove every edge $uv \in E$ and add an edge-vertex e_{uv} adjacent to u and v (see Figure 5.13). Notice that, at time step 1, there are only $n-k$ firefighters that can be placed freely because of the guard vertices.

We claim that (G, k) is a yes-instance if and only if (G', s, k_b, b) is a yes-instance.

"⇒": Suppose that we have a clique $C \subseteq V$ of size k and consider the following strategy. At time step one, the strategy uses the $n-k$ remaining firefighters to protect all the original vertices V in G' except those in the clique C. At time step two, all the k vertices of C are burned. Since there are $n-k$ guard vertices attached to each vertex in C, we need to protect $b_1 = k(n-k)$ vertices. Moreover, there are $k\Delta - \binom{k}{2}$ edge-vertices adjacent to the vertices in the clique C. Since there remain $b_2 = b - b_1$ firefighters, we can protect all of them. Hence, no more than $k + 1 = k_b$ vertices are burned at the end of the process.

"⇐": Conversely, suppose that there is no clique of size k in G. At time step one, any valid strategy has to place the $n-k$ remaining firefighters on vertices that are not edge-vertices; otherwise at least $k + 2 > k_b$ vertices will burn. At time step two, since there is no clique of size k, there will be at least $k\Delta - \binom{k}{2} + 1$ edge-vertices adjacent to the k burned vertices. For the same reason as before, there remains $b_2 = b - b_1$ firefighters which is not enough to protect these edge-vertices. Therefore, given any valid strategy there will be at least $k + 2 > k_b$ burned vertices.

We note that the parameters used in this reduction in are linearly related. Since Clique on regular graphs is W[1]-hard with respect to k and cannot be solved in time $n^{o(k)}$ unless ETH fails [86], we obtain the desired lower bound $n^{o(k_b)}$. ■

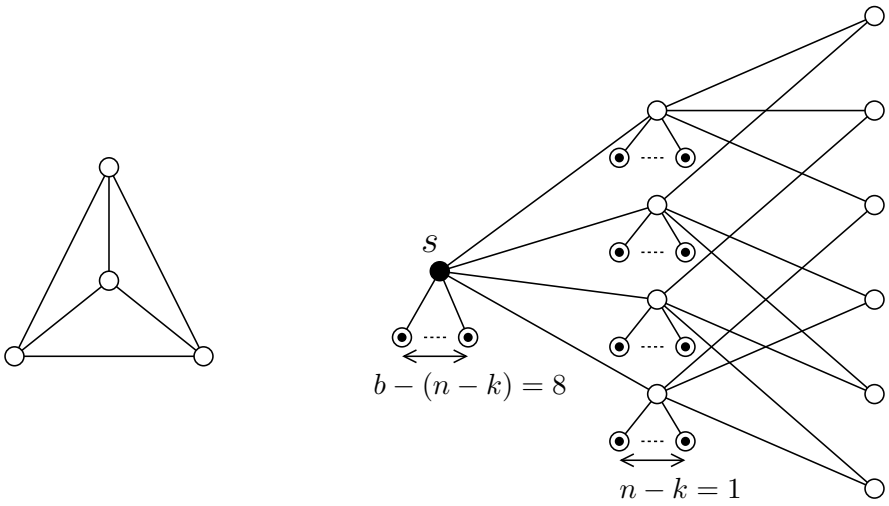


Figure 5.13: Construction of G' (right) from a regular graph of degree three (left). In this example, we have $k = 3$, $n = 4$, $\Delta = 3$, $b_1 = 3$, $b_2 = 6$ and $b = 9$. Guard vertices are represented by a dot within a circle.

5.5 Parameterized algorithms

In this section, we start by showing the fixed-parameter tractability of Dual Firefighter with respect to the combined parameter k_b and b. Then, we resolve several open questions of Cai et al. [27] on trees. We also refine and extend some of their results. Finally, we extend these results to the graphs of bounded treewidth. We further derive that Firefighter is in FPT with respect to the parameter k on graphs of bounded local treewidth, including planar graphs, graphs of bounded genus, apex-minor-free graphs, and graphs of bounded maximum vertex degree.

5.5.1 Dual Firefighter parameterized by k_b and b

In contrast to what we have previously shown, we prove that Dual Firefighter becomes fixed-parameter tractable if we combine the parameter k_b with the budget b.

Theorem 84 Dual Firefighter *is fixed-parameter tractable with respect to the combined parameter k_b and b. It can be solved in $O((2^{b+1} - 1)^{k_b+b-1}n)$ time.*

Proof. In this proof, we consider the constrained protection phase. We present a simple branching algorithm. Assume that we are in the i-th time step and let B be the set of vertices which are currently on fire. Moreover, let P be the set of already protected vertices (in the first step we have $B = \{s\}$ and $P = \emptyset$). Let $a = ib - |P|$ and $r = |N(B) \setminus P|$. The algorithm does the following:

1. If $|B| > k_b$, then we immediately answer "no".
2. Observe that in the i-th step we are allowed to protect at most $\min(a, r)$ vertices.

If $a \geq r$, then we can greedily protect the whole set $N(B) \setminus P$. Hence in this case we answer "yes".

3. In the last case, when $a < r$, we branch on all subsets of $N(B) \setminus P$ of size at most a. Observe that the number of branches is equal to $\sum_{j=0}^{a} \binom{r}{j} \leq 2^r - 1$, since we have $a < r$.

The running time of the algorithm is as follows. We introduce a measure $\alpha = (k_b - |B|) + (ib - |P|)$ which we use in our time bound. At the beginning of the first step of the burning process, we have $\alpha = (k_b - 1) + (b - 0) = k_b + b - 1$. By $T(\alpha)$ we denote the upper bound on the number of steps that our algorithm requires for a graph with measure value α. Observe that for $\alpha \leq 0$, we have $T(\alpha) = O(n)$. Let us assume that the algorithm did not stop in step 1 nor 2, and it branches into at most $2^r - 1$ choices of protected vertices. Observe that no matter how many vertices the algorithm decides to protect, the value of the measure decreases by exactly $r - b$. Consequently, we have the inequality $T(\alpha) \leq (2^r - 1)T(\alpha - r + b) + O(n)$. Since the algorithm did not stop in steps 1 or 2, we infer that $r \geq b+1$. The time bound follows from the fact that the worst case for the inequality occurs when $r = b + 1$. ■

Using an easy observation, we can deduce the following corollary.

Corollary 85 Dual Firefighter *is fixed-parameter tractable with respect to the combined parameter k_b and "maximum degree" Δ of the input graph. It can be solved in $O((2^{\Delta+1} - 1)^{k_b+\Delta-1}n)$ time.*

Proof. Let (G, s, b, k_b) be an instance of Dual Firefighter where G has maximum degree Δ. If $b \geq \Delta$ then protect all the vertices in $N(s)$ at time step one. Otherwise, apply the algorithm from Theorem 84 that runs in $O((2^{\Delta+1} - 1)^{k_b+\Delta-1}n)$ since $b < \Delta$. ■

5.5.2 Firefighting on trees

In this section, we study the Firefighter, Bounded Firefighter and Dual Firefighter problems on trees. Previously known results on that class of graphs are from Cai et al. [27]. More specifically, they obtained the following parameterized algorithms on trees and budget $b = 1$.

- A randomized algorithm solving Firefighter in time $O(4^k + n)$, which can be derandomized to a $O(n + 2^{O(k)})$-time algorithm. They also gave a polynomial kernel with respect to the parameter k.

- A randomized algorithm for Dual Firefighter of running time $O(4^{k_b}n)$, which can be derandomized to a $O(2^{O(k_b)}n \log n)$-time algorithm. They left as an open problem whether Dual Firefighter has a polynomial kernel with respect to the parameter k_b.

- For Bounded Firefighter, they gave a randomized algorithm of running time $O(k_p^{O(k_p)}n)$, which can be derandomized to a $O(k_p^{O(k_p)}n \log n)$-time algorithm. They left open whether the problem has a polynomial kernel with respect to the parameter k_p, and asked whether there is an algorithm solving the problem in time $2^{o(k_p \log k_p)}n^{O(1)}$.

In what follows, we give a deterministic $O((b+1)^{k_b}n)$-time algorithm for Dual Firefighter on trees which improves the $O(4^k n)$ running time of the randomized algorithm from [27] for $b = 1$. We also answer the open question of Cai et al. by showing that Dual Firefighter parameterized by k_b has no polynomial kernel on trees of maximum vertex degree four for $b = 1$, and no polynomial kernel on trees of maximum vertex degree $b + 4$ for any $b \geq 2$.

We provide a deterministic algorithm solving Bounded Firefighter on trees in $O((b + 1)^{k_p/b+1}k_p n)$ time. This answers an open question of Cai et al. since it gives us a $O(2^{k_p}k_p n)$-time algorithm when $b = 1$. Furthermore, we show that the problem has no polynomial kernel with respect to the parameter k_p on trees. This last result was independently obtained by Yang [104]. Based on the parameterized algorithm, we also give an exact subexponential-time algorithm, for the classical firefighter problem (*i.e.* the Firefighter problem with budget $b = 1$) on a tree in $O((b + 1)^{\sqrt{2n/b}}n^{3/2})$ time, thus improving on the $2^{O(\sqrt{n}\log n)}$ running time from [27, 56] for $b = 1$.

Finally, we show a deterministic algorithm solving Firefighter on trees in $O((b + 1)^{k/b+3}kn)$ time, improving the running time $O(4^k + n)$ of the randomized algorithm from [27] for $b = 1$.

Dual Firefighter. Using a simple branching algorithm we get the following result.

Theorem 86 Dual Firefighter *is solvable in* $O((b+1)^{k_b}n)$ *time on trees.*

Proof. Let (T, s, b, k_b) be an instance of Dual Firefighter where T is a tree with root s. If s has at most b children, then we immediately answer "yes". We may assume that the root has exactly a children where $a < b + k_b$ since otherwise we simply answer "no". We use Lemma 58 and branch on every subset of b children of the root s. In each branch, we cut the subtree rooted at the protected vertex, identify all the vertices that are on fire after the first step, and decrease the parameter by $a - b$. In this way, we obtain a new instance of the Dual Firefighter problem with parameter value equal to $k_b - (a - b)$. The time bound follows from the inequality

$$T(k_b) \leq \binom{a}{b} T(k_b - (a - b)) + O(n)$$

which is worst when $a = b + 1$. ■

The above theorem directly implies that Dual Firefighter admits a kernel of size $(b+1)^{k_b}$. However, we show in the following that this size cannot be reduced to $(b+k_b)^{O(1)}$ unless NP $\subseteq$ coNP/poly. As a first step toward this goal, we provide the proof for $b = 1$ and then extend this result to any fixed $b \geq 1$.

Theorem 87 *There is no polynomial kernel for* Dual Firefighter *with respect to the parameter* k_b *and budget one even for trees of maximum degree four unless* NP $\subseteq$ coNP/poly.

Proof. We apply Theorem 4 (see subsection 2.3.2), where as the language L we use Dual Firefighter on trees of maximum degree three and budget one, which is NP-complete [55]. In the following, we show that Dual Firefighter on trees with maximum degree three and budget one cross-composes to Dual Firefighter parameterized by k_b on trees with maximum degree four.

Observe that any polynomial equivalence relation is defined on all words over the alphabet Σ and for this reason we should also define how the relation behaves on words that do not represent instances of the problem. For the equivalence relation $\mathcal{R}$ we take a relation that puts all malformed instances into one equivalence class and all well-formed instances are grouped according to the number of vertices we are allowed to burn.

If we are given malformed instances, we simply output a trivial no-instance. Thus in the rest of the proof we assume we are given a sequence of instances $(T_i, s_i, 1, k_b)_{i=1}^{t}$ of Dual Firefighter, where each T_i is of maximum degree three. Observe that in all instances we have the same value of the parameter k_b. Without loss of generality, we assume that $t = 2^h$ for some integer $h \geq 1$. Otherwise we can duplicate an appropriate number of instances $(T_i, s_i, 1, k_b)$.

We create an instance $(T, s, 1, k'_b)$ of Dual Firefighter parameterized by k_b where T is a tree defined as follows. We start with a full binary tree rooted at a vertex s with exactly t leaves . Now for each $i = 1, \ldots, t$, we replace the i^{th} leaf of the tree by the tree T_i rooted at s_i. Finally, we set $k'_b = k_b + h = k_b + \log_2 t$. Observe that since each tree T_i is of maximum degree three, the tree T is of maximum degree four. To prove correctness, it is enough to show that any strategy that minimizes the number of burned vertices protects exactly one vertex at each level $1, \ldots, h$, which follows from Lemma 58. Hence in any strategy that minimizes the number of burned vertices, there will be exactly one vertex s_i which is on fire after h steps. ■

We now generalize the above result using the fact that Dual Firefighter is NP-complete for trees of maximum degree $b + 3$ where $b \geq 2$ is fixed (Theorem 61).

Theorem 88 *For any fixed budget $b \geq 2$, there is no polynomial kernel for the* Dual Firefighter *with respect to the parameter k_b even for trees of maximum degree $b+4$ unless* NP $\subseteq$ coNP/poly.

Proof. Let $b \geq 2$ be any fixed constant. We use Theorem 4, where L is the Dual Firefighter problem on trees of maximum degree $b + 3$ and budget b. To give the cross-composition, we use the same proof from Theorem 87 with the following differences.

- For the equivalence relation $\mathcal{R}$, we take a relation that puts all malformed instances into one equivalence class, and all well-formed instances are grouped according to the parameter value k_b and the budget b.
- We assume that $t = (b + 1)^h$ for some integer $h \geq 1$.
- The tree T is now constructed from a full $(b + 1)$-ary tree rooted at vertex s with exactly t leaves. Finally, we set $k'_b = k_b + \log_{b+1} t$.

This completes the proof. ■

Bounded Firefighter. Before we present the algorithm for this problem, we need to introduce the following notation. Let $T = (V, E)$ be any rooted tree with n vertices. Use a pre-order traversal of T to number the vertices of T from 1 to n. We say that $u \in V$ is *to the left* of $v \in V$ if the number assigned to u is not greater than the number of v in the order. It is then easy to define what the *leftmost* or *rightmost* vertex is.

As a matter of fact we provide an algorithm for solving the optimization problem Max Bounded Firefighter. Obviously, this implies that Bounded Firefighter is

also solvable with the same running time by applying this algorithm.

Theorem 89 MAX BOUNDED FIREFIGHTER *is solvable in* $O((b+1)^{(k_p/b)+1}k_p n)$ *time on trees.*

Proof. Let (T, s, b, k_p) be an instance of MAX BOUNDED FIREFIGHTER on a tree T. Assume that T is rooted at s and let $h = \lceil k_p/b \rceil$. By Lemma 58, we can define a characteristic vector χ_v of length h for each vertex v of the tree, which has value $p_i \in \{0, \ldots, b\}$ at position i if and only if the optimal strategy protects p_i vertices at level i in the part of the tree to the left of v. We use these vectors as the basis for a dynamic programming procedure. However, the vector cannot ensure that no ancestors of a protected vertex will be protected. To ensure this, we add another dimension to our dynamic programming procedure. The pre-order numbering ensures that no descendant is protected.

The dynamic programming algorithm is then as follows. Let L be the set of vertices in T that are at level at most h. For each $v \in L$, let P_v denote the path in T between v and s. For each vector $\chi \in \{0, \ldots, b\}^h$ and each integer $0 \leq i \leq h$, we compute $A_v(\chi, i)$, the maximum number of vertices one can save when protecting at most $\chi(j)$ vertices at level j, where protected vertices must lie to the left of v but at level greater than i when lying on P_v, and no protected vertex is an ancestor of another. Observe that s is the leftmost vertex of L. Now set $A_s(\chi, i) = 0$ for any χ and i. Then

$$\begin{aligned} A_v(\chi, i) \quad = \max \{ \quad & A_{l(v)}(\chi, \min\{\text{level}(v) - 1, i\}), \\ & [\chi(\text{level}(v)) \geq 1 \wedge \text{level}(v) > i] \cdot \\ & (r(v) + A_{l(v)}(\chi^v, \text{level}(v) - 1)) \ \} \end{aligned}$$

Here level(v) is the level where vertex v is lying, $l(v)$ is the rightmost vertex in L which has strictly smaller value in the pre-order than v, and $r(x)$ is the number of vertices saved when protecting only x. Moreover, χ^v is the vector obtained from χ by reducing the number $p_{\text{level}(v)}$ by 1. In the formula we use Iverson's bracket notation, where $[\phi]$ is equal to one if ϕ is true and zero otherwise.

To see that the above formula is correct, observe that we can either protect the considered vertex v or not. If we do not protect v, then we must ensure that the value for the second dimension of our dynamic programming procedure does not exceed the length of P_v, yet still captures the same forbidden part of P_v. Correctness then follows from the fact that the parent of v is always on $P_{l(v)}$. If we do protect v, we can protect v only if we are allowed to do so, *i.e.* if $\chi(\text{level}(v)) \geq 1$ and $\text{level}(v) > i$. Furthermore, we need to ensure that no ancestor of v is protected later. Therefore, we set the value for the second dimension of our dynamic programming procedure to level$(v) - 1$.

To get the solution for the whole tree T, return $A_{v^*}(\chi^*, 0)$, where v^* is the rightmost vertex of L and χ^* is a vector of length h with the h-th entry set to $k_p - (h-1)b$ and the other entries set to b. To obtain the claimed running time, first find L, and then $l(v)$ for each vertex $v \in L$. This can be done in linear time by a depth-first search. We can also compute $r(x)$ for each $x \in V(T)$ in linear time, as $r(x)$ equals one plus the number of descendants of x. By traversing the vertices of L from left to right, the total running time is $O((b+1)^h k_p n) = O((b+1)^{(k_p/b)+1} k_p n)$. ■

Since we assumed that $b \leq k_p$ (see the discussion in Section 5.2), we have the following

immediate corollary.

Corollary 90 Bounded Firefighter *is fixed-parameter tractable with respect to the parameter k_p on trees.*

The following result states that it is unlikely that Bounded Firefighter parameterized by k_p admits a polynomial kernel on trees. Like we did for Dual Firefighter, we give the proof for $b = 1$ first and then extend it for any fixed $b \geq 2$.

Theorem 91 *There is no polynomial kernel for* Bounded Firefighter *with respect to the parameter k_p and budget one even for trees of maximum degree four unless* NP $\subseteq$ coNP/poly.

Proof. There are only three differences compared to the proof for Theorem 87.

- The language L is now Bounded Firefighter on trees of maximum degree three and budget one.
- For the equivalence relation $\mathcal{R}$, we take a relation that puts all malformed instances into one equivalence class, and all well-formed instances are grouped according to the number of vertices of the tree, the parameter value k_p, and the value k.
- The instance $(T, s, 1, k', k'_p)$ of Bounded Firefighter parameterized by k_p is as follows. The tree T is constructed as described in the proof of Theorem 87. The value k'_p is set to $k_p + h$, and the value of k' is equal to $k + (t-1)n + (t-h-1)$, where n is the number of vertices in each of the trees T_i. The additional summands are derived from the fact that any optimal strategy will ensure that after h steps exactly one vertex s_i will be on fire and hence we save $t-1$ subtrees rooted at s_i, each containing n vertices, and $t-h-1$ vertices of the full binary tree.

This completes the proof. ■

Using the fact that Theorem 61 implies the NP-completeness of Bounded Firefighter for trees of maximum degree $b+3$ for any fixed budget $b \geq 2$, we may generalize the previous result as follows.

Theorem 92 *For any fixed budget $b \geq 2$, there is no polynomial kernel for* Bounded Firefighter *with respect to the parameter k_p even for trees of maximum degree $b+4$ unless* NP $\subseteq$ coNP/poly.

Proof. We start with the proof from Theorem 91 and modify it as follows.

- For the equivalence relation $\mathcal{R}$, we take a relation that puts all malformed instances into one equivalence class, and all well-formed instances are grouped according to the number of vertices of the tree, the parameter value k_p, the value k and the budget b.
- We assume that $t = (b+1)^h$ for some integer $h \geq 1$.
- The tree T is now constructed from a full $(b+1)$-ary tree rooted at vertex s with exactly t leaves.

This completes the proof. ■

Firefighter. Using the dynamic programming algorithm of Theorem 89, we can easily prove the following.

Corollary 93 FIREFIGHTER *is solvable in* $O((b+1)^{(k/b)+2}kn)$ *time on trees.*

Proof. Let (T, s, b, k) be an instance of FIREFIGHTER on a tree T. Using the same argument from Theorem 76, we run the dynamic programming algorithm for MAX BOUNDED FIREFIGHTER (see Theorem 89) for all $k_p = 1, \ldots, k$. Furthermore, we note that

$$\sum_{i=1}^{k}((b+1)^{(i/b)+1}in) \;\leq\; kn(b+1)\sum_{i=1}^{k}(b+1)^{i/b} \;\leq\; (b+1)^{k/b+2}kn,$$

implying that the worst-case running time is $O((b+1)^{k/b+2}kn)$. ■

As we assumed that $b \leq k$ (see the discussion in Section 5.2), we get the following immediate corollary.

Corollary 94 FIREFIGHTER *is fixed-parameter tractable with respect to the parameter* k *on trees.*

A subexponential algorithm. To obtain a good subexponential algorithm for the MAX FIREFIGHTER problem, we use the following lemma. A similar idea for $b = 1$ appeared independently in [56].

Lemma 95 *If a vertex at level* ℓ *burns in an optimum strategy for an instance of the* MAX FIREFIGHTER *problem on trees, then at least* $\frac{b}{2}(\ell^2+\ell)$ *vertices are saved.*

Proof. Let (T, s, b) be an instance of the MAX FIREFIGHTER problem on trees, and let v be a vertex of level ℓ that burns in an optimum strategy. Then the strategy protects b vertices at level ℓ, and by Lemma 58 it thus protects b vertices $p_i^1, \ldots, p_i^b$ at each level i for $1 \leq i \leq \ell$. For any i, each of the subtrees rooted at $p_i^1, \ldots, p_i^b$ should contain at least $\ell - i + 1$ vertices, or it would have been better to protect the vertex at level i that is on the path from v to s. But then the strategy saves at least $b\sum_{i=1}^{\ell}(\ell - i + 1) = \frac{b}{2}(\ell^2+\ell)$ vertices. ■

Theorem 96 MAX FIREFIGHTER *is solvable in* $O((b+1)^{\sqrt{2n/b}}n^{3/2})$ *time on trees.*

Proof. Let (T, s, b) be an instance of the MAX FIREFIGHTER problem on trees. Suppose that a vertex v at level $\sqrt{2n/b}$ burns in an optimum strategy. Then, by Lemma 95, the strategy saves at least $n + \sqrt{bn/2} > n$ vertices, which is not possible. It follows that all vertices at level $\sqrt{2n/b}$ are saved in any optimum strategy. Since in any optimum strategy every protected vertex has a burned ancestor by Lemma 58, all protected vertices are at level at most $\sqrt{2n/b}$. Hence there is an optimum strategy that protects at most $b\sqrt{2n/b} = \sqrt{2bn}$ vertices, and we can find the optimum strategy by running the algorithm of Theorem 89 with $k_p = \sqrt{2bn}$. ■

This implies an $O(2^{\sqrt{2n}}n^{3/2})$-time algorithm for the classical firefighter problem ($b = 1$) on trees.

5.5.3 Firefighting on tree-like graphs

We generalize the above results by showing that Bounded Firefighter (resp. Firefighter) is fixed-parameter tractable when parameterized by k_p (resp. k) and the treewidth of the underlying graph. To this end, we use Monadic Second Order Logic (see Section 2.3.1). As a matter of fact, we show that Max Bounded Firefighter can be solved in $f(\mathrm{tw}, k_p)\cdot n^{O(1)}$ time where tw is the treewidth of the input graph. To do that, we actually need Linear Extended MSOL (LEMSOL) [10], which allows the maximization over a linear combination of the size of unbound set variables in the MSOL formula. (The definition of LEMSOL in [10] is slightly more general, but this suffices for our purposes.)

Theorem 97 Max Bounded Firefighter *is solvable in* $f(\mathrm{tw}, k_p)\cdot n^{O(1)}$ *time where* tw *is the treewidth of the input graph and* f *is a function that solely depends on* tw *and* k_p.

Proof. Let (G, s, b, k_p) be an instance of Max Bounded Firefighter such that the treewidth of G is tw. Recall that we assumed $b \leq k_p$ (see the discussion in Section 5.2). Use Bodlaender's Algorithm [22] to find a tree decomposition of G of width at most tw. Consider the following MSOL formulae.

$$\begin{aligned}&\mathrm{NextBurn}(B_{i-1}, B_i, p_1^1, \ldots, p_1^{b_1}, \ldots, p_i^1, \ldots, p_i^{b_i}) \ :=\\ &\quad \forall v \left(\left(v \in B_{i-1} \vee \exists u \left(u \in B_{i-1} \wedge \mathrm{adj}(u, v) \wedge \left(\bigwedge_{\substack{1\leq j\leq i\\ 1\leq h\leq b_j}} v \neq p_j^h \right) \right) \right) \Leftrightarrow v \in B_i \right)\end{aligned}$$

This expresses that if the vertices of B_{i-1} are burning by time step $i-1$, then the vertices of B_i burn by time step i, assuming that vertices $p_1^1, \ldots, p_1^{b_1}, \ldots, p_i^1, \ldots, p_i^{b_i}$ have been protected so far, with $b_j \leq b$ for $j = 1, \ldots, i$.

$$\begin{aligned}&\mathrm{Saved}(S, B, p_1^1, \ldots, p_1^{b_1}, \ldots, p_m^1, \ldots, p_m^{b_m}) \ :=\\ &\quad \forall u \Big(u \in S \Rightarrow \Big(u \notin B \wedge \forall v \Big(\mathrm{adj}(u, v) \Rightarrow v \in S \vee \bigvee_{\substack{1\leq i\leq m\\ 1\leq h\leq b_i}} p_i^h = u \Big) \Big) \Big)\end{aligned}$$

This expresses that S is a set of saved vertices when B is a set of burned vertices and vertices $p_1^1, \ldots, p_1^{b_1}, \ldots, p_m^1, \ldots, p_m^{b_m}$ are protected, with $b_j \leq b$ for $j = 1, \ldots, m$.

$$\begin{aligned}\mathrm{Protect}(S, b_1, \ldots, b_\ell) := \exists p_1^1, \ldots, p_1^{b_1}, \ldots, p_\ell^1, \ldots, p_\ell^{b_\ell} \ \exists B, B_0, \ldots, B_{\ell-1}&\\ \forall u \ (u \in B_0 \Leftrightarrow u = s) &\quad (5.3)\\ \wedge \bigwedge_{1\leq i\leq \ell-1} \mathrm{NextBurn}(B_{i-1}, B_i, p_1^1, \ldots, p_1^{b_1}, \ldots, p_i^1, \ldots, p_i^{b_i}) &\quad (5.4)\\ \wedge \bigwedge_{\substack{1\leq i\leq \ell\\ 1\leq h\leq b_i}} p_i^h \notin B_{i-1} &\quad (5.5)\\ \wedge \forall u \left(\Big(\bigvee_{0\leq i\leq \ell-1} u \in B_i \Big) \Rightarrow u \in B \right) &\quad (5.6)\\ \wedge \mathrm{Saved}(S, B, p_1^1, \ldots, p_1^{b_1}, \ldots, p_\ell^1, \ldots, p_\ell^{b_\ell}) &\quad (5.7)\end{aligned}$$

This expresses that S can be saved by protecting b_i vertices in step i. The sets B_i contain all vertices that are burned by time step i, which is ensured by the formulas in lines 5.3 and 5.4. The set B contains vertices that are not saved (line 5.7) and all vertices of the sets B_i (line 5.6). The vertices $p_1^1, \ldots, p_1^{b_1}, \ldots, p_\ell^1, \ldots, p_\ell^{b_\ell}$ are the vertices that are protected. Line 5.5 ensures that the vertices we want to protect are not burned by the time we pick them. Then we want to find the largest set S such that

$$\text{Protect}_{k_p}(S) := \bigvee_{1 \leq \ell \leq \lceil k_p/b \rceil} \bigvee_{\substack{1 \leq d \leq b \\ b(\ell-1)+d \leq k_p}} \text{Protect}(S, \overbrace{b, \ldots, b}^{\ell-1}, d)$$

is true. Following a result of Arnborg, Lagergren, and Seese [10], this can be done in $f(\text{tw}, k_p) \cdot n^{O(1)}$ time using the above formula. ■

From the above Theorem, we immediately deduce the following.

Corollary 98 Bounded Firefighter *is fixed-parameter tractable with respect to the combined parameter k_p and treewidth of the input graph.*

In the same way as Corollary 93, we then obtain the following.

Corollary 99 Firefighter *is fixed-parameter tractable with respect to the combined parameter k and treewidth of the input graph.*

This algorithm also works on graphs of bounded local treewidth.

Corollary 100 Firefighter *is fixed-parameter tractable with respect to the parameter k on graphs of bounded local treewidth.*

Proof. Let (G, s, b, k) be an instance of Firefighter. Observe that if the graph G has a vertex at distance more than k from the initially burned vertex s, then a strategy that protects any vertex at distance i from s in time step i will save at least k vertices, and we can answer "yes" immediately. From the previous discussion, we may assume that $G = G[N^k(s)]$. Because G has bounded local treewidth, there exists a function f such that $\text{ltw}_k(G) \leq f(k)$. Therefore, we have $\text{tw}(G) \leq f(k)$ since $G = G[N^k(s)]$ and thus $\text{tw}(G) = \text{ltw}_k(G)$. ■

The class of graphs having bounded local treewidth coincides with the class of apex-minor-free graphs [53], which includes the class of planar graphs.

Corollary 101 Firefighter *is fixed-parameter tractable with respect to the parameter k on planar graphs.*

5.6 Parameter "vertex cover number"

In this section, we show that the Firefighter problem is fixed-parameter tractable with respect to the parameter "vertex cover number".

Theorem 102 Firefighter *admits a kernel with $O(2^\tau \tau^2)$ vertices where τ is the vertex cover number of the input graph.*

Proof. Since Firefighter fpt-reduces with respect to the parameter vertex cover number to Dual Firefighter by setting $k_b = n - k$, we will prove the result for Dual Firefighter for convenience.

Let $(G = (V, E), s, b, k_b)$ be an instance of Dual Firefighter with $|V| = n$. We may assume that $|N(s)| < b + k_b$, because otherwise the answer is clearly "no". First, we compute in polynomial-time a 2-approximate vertex cover $C' \subseteq V$ where $|C'| = \tau' \leq 2\tau$.

If $b \geq \tau'$ then the answer is "yes". Indeed, suppose that $b \geq \tau'$. If $s \notin C'$, then it is enough to protect all the vertices in C' at the first step to stop the fire. If $s \in C'$, then let $N_1(s) = N(s) \cap C'$ and $N_2(s) = N(s) \setminus C'$. If $|N_2(s)| < k_b$, then the strategy that protects all the vertices in C' at the first step will result in at most k_b burned vertices in G. If $|N_2(s)| \geq k_b$, then consider the following strategy. At the first time step, protect all the vertices in $N_1(s)$, and $b - |N_1(s)|$ vertices of $N_2(s)$. Notice that $|N_1(s)| \leq b$ since $|N(s)| \leq b + k_b$. At the second time step, protect all the vertices in $C' \setminus N_1[s]$. Hence, the number of burned vertices using this strategy is at most k_b.

From now on, we assume $b < \tau'$. Observe that every two steps at least one vertex inside C' gets burned. It follows that after $2\tau'$ steps, all the vertices in G are necessarily burned. Since the number of steps is at most $2\tau'$, there is a total of at most $2\tau' b < 2\tau'^2$ protected vertices.

We now provide the kernelization algorithm. We call a set $T \subseteq V$ a *twins set* if for every $v, u \in T$, $v \neq u$, we have $N(u) = N(v)$ and $uv \notin E$. Consider the following reduction rule.

Reduction rule. If there exists a twins set $T \subseteq V \setminus C'$ such that $|T| \geq 2\tau'^2 + 1$, then delete $|T| - 2\tau'^2 - 1$ vertices of T.

As to the correctness of the rule, let Φ be a strategy such that at most k_b vertices are burned in G and $T \subseteq V \setminus C'$ be a twins set. Observe that if Φ protects a subset $T_1 \subseteq T$ of vertices then protecting any subset $T_2 \subseteq T$ instead of T_1 such that $|T_2| = |T_1|$ leads to another strategy that saves exactly the same number of vertices. It follows that if Φ protects a vertex in a twins set that has been deleted by the reduction rule then we can protect any other non-deleted vertex in the same twins set instead, without changing the amount of saved vertices. Moreover, we have $|T_1| \leq 2\tau'^2$ since at most $2\tau'^2$ vertices are protected. Therefore, it is enough to keep $2\tau'^2 + 1$ vertices in each twins set of $V \setminus C'$. Conversely, if there is a strategy for which at most k_b vertices in G' are burned, then applying this strategy in G will result in at most k_b burned vertices in G.

Let $G' = (V', E')$ be the graph obtained by iteratively applying the above reduction rule to every twins set in $V \setminus C'$. Notice that the procedure runs in polynomial time. The number of distinct twins sets in $V \setminus C'$ is at most 2^τ (one for each subset of C'). Moreover, each of these twins set has at most $2\tau'^2 + 1$ vertices. Therefore, the size of the reduced instance is $O(2^\tau \tau^2)$. This completes the proof. ■

While the above result is useful as a complexity classification result, it might be more efficient to use the following kernelization instead, depending of the size of τ and k.

Theorem 103 Firefighter *admits a kernel with $O(2^\tau k)$ vertices where τ is the vertex cover number of the input graph.*

Proof. We use the same approach as in the proof of Theorem 102 with the following differences. Let (G, s, b, k) be an instance of Firefighter. We iteratively apply the

following reduction rule to every twins set in G to obtain a new graph $G' = (V', E')$.

Reduction rule: If there exists a twins set T such that $|T| \geq k+1$, then delete $|T| - k - 1$ vertices of T.

Let $C \subseteq V'$ be a minimum vertex cover and let $D = V' \setminus C$ be an independent set. The number of distinct twins sets in D is at most 2^τ (one for each subset of C). Moreover, each twins set in G' has at most $k+1$ vertices. Therefore, the size of the reduced instance is at most $O(2^\tau k)$.

The proof of correctness uses similar arguments as for Theorem 102 since we may assume that no strategy can protect more that k vertices otherwise the problem is trivial. ■

5.7 Approximability

Finbow and MacGillivray [54] asked whether there exists a constant $c > 1$ such that the degree greedy algorithm that consists, at each time step, to protect a highest degree vertex adjacent to a burning vertex, gives a polynomial-time c-approximation for Max Firefighter for trees. The following proposition answers this question negatively.

Theorem 104 *For any budget $b \geq 1$, there exists no function $\alpha : \mathbb{N} \to (1, n)$ such that the degree greedy algorithm is a polynomial-time $\alpha(n)$-approximation algorithm for* Max Firefighter *for trees.*

Proof. Consider a tree $\mathcal{T}(r, h-1, b+1)$ where h is a positive integer (see definition in the proof of Theorem 61). Add a vertex s adjacent to r and a vertex v_1 adjacent to s; for $i = 2, \ldots, h-1$, add a path of length $i-1$ with endpoints u_i and v_i such that u_i is adjacent to s; finally, for $i = 1, \ldots, h-1$, add $b+2$ vertices adjacent to v_i (see Figure 5.14).

Notice that the degree greedy algorithm protects vertices in the following order: $v_1, \ldots, v_{h-1}$. Thus it saves $g_h = (h-1)(b+2)$ vertices. However, it is not difficult to see that the optimal solution protects vertices in the following order: $r, v_2, \ldots, v_{h-1}$. Thus, in an optimal solution we save $opt_h = (h-2)(b+2) + \sum_{i=0}^{h-1}(b+1)^i$ vertices. Since $\frac{opt_h}{g_h} \to +\infty$ when $h \to +\infty$, the result follows. ■

In contrast to Max Firefighter which is constant approximable on trees, the following theorem shows a strong inapproximability result for Min Firefighter even when restricted to trees.

Theorem 105 *For any $\varepsilon > 0$ and any budget $b \geq 1$,* Min Firefighter *is not $n^{1-\varepsilon}$-approximable even for trees, unless $P = NP$.*

Proof. We construct a simple gap-introducing reduction from Save to Min Firefighter. Let $I = (T, C, s, b)$ be an instance of Save consisting of a tree $T = (V, E)$ with $|V| = n_1$ and a subset $C \subseteq V$ which corresponds to the set of leaves. We construct an instance $I' = (T', s', b)$ of Min Firefighter consisting of a tree $T' = (V', E')$ with $|V'| = n$ as follows. For every leaf ℓ of T, add $\lfloor n_1^\beta + b \rfloor$ vertices adjacent to ℓ where $\beta = \frac{4}{\varepsilon} - 3$. Notice that $n = \lfloor n_1^\beta + b \rfloor |S| + n_1 < n_1^{\beta+3}$.

If there exists a strategy that saves all the vertices in C then at most n_1 vertices are burned in V'.

Conversely, if there is no strategy that saves all the vertices in C then at least $n_1^\beta >$

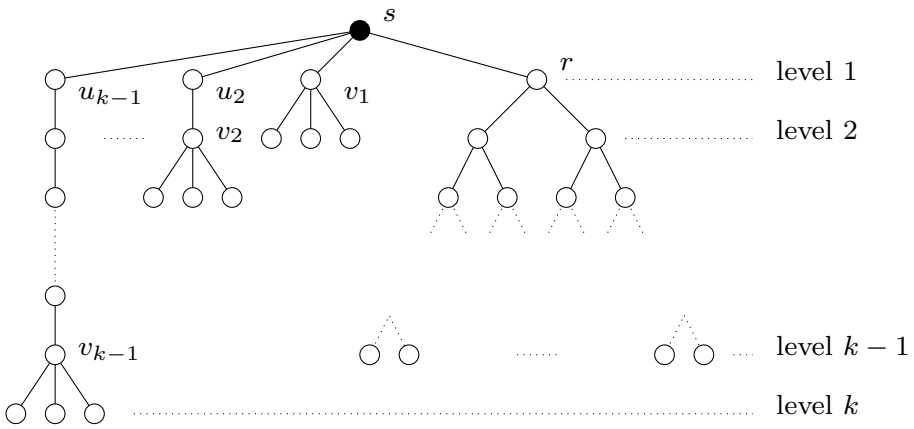


Figure 5.14: Instance where the degree greedy algorithm gives no approximation guarantee for the case $b = 1$. Since here $S = V$, we did not represent vertices in S by squares.

> $n^{1-\varepsilon}n_1$ vertices are burned in V'. Using Theorem 8, we deduce that there is no polynomial-time $n^{1-\varepsilon}$-approximation algorithm for Min Firefighter unless P = NP. ■

5.8 Conclusion and open problems

In this chapter, we studied some generalizations and variants of the firefighter problem when more than one firefighter is available at each time step. We would like to conclude this study with the following research directions.

(1) The complexity of Firefighter and Save in the case when the number of firefighters at each time step depends on the number of vertices. The complexity of Firefighter for trees of maximum degree $b + 2$ is not established.

(2) The Max Firefighter problem is 2-approximable for trees. Establishing non approximability results or better approximability results is an interesting open question.

(3) We showed that the Firefighter problem is NP-complete for graphs of pathwidth at most three. While it is polynomial-time solvable for k-caterpillars, we do not know whether this algorithm can be extended to general graphs of pathwidth two.

(4) We have shown that Firefighter parameterized by k is in FPT on graphs of bounded local treewidth, and thus on planar graphs, by showing that the problem is in FPT parameterized by k and the treewidth of the input graph. While Bounded Firefighter parameterized by k_p is also in FPT parameterized by k_p and the treewidth, we do not know whether the problem is in FPT on planar graphs, and leave it as an open problem.

(5) We do not know if Dual Firefighter is fixed-parameter tractable when parameterized by k_b and the treewidth. One could try to adapt the MSOL formula for Max Bounded Firefighter, but in its current form the size of the formula depends on k_b and b. This is not the case for Theorem 97, as we can assume that $b \leq k_p$. However, it is not clear whether a similar assumption can be made for parameter k_b.

(6) The Max Firefighter problem is solvable in subexponential time on trees. Is it solvable in time $2^{o(n)}$ on planar graphs? Even the case of outerplanar graphs is open.

(7) It is unknown if Firefighter admits a polynomial kernel for parameter "vertex cover number". While we have shown that Bounded Firefighter and Dual Fire-

FIGHTER do not admit a polynomial kernel with respect to the combined parameter k_p and tw, and k_b and tw, respectively, we do not know if it is also the case for FIREFIGHTER parameterized by k and tw.

(8) We have proved that FIREFIGHTER and DUAL FIREFIGHTER have no $n^{o(\sqrt{k})}$ and $n^{o(\sqrt{k_b})}$ algorithms, respectively, unless ETH fails. However, the existence of $n^{o(k)}$- and $n^{o(k_b)}$-time algorithms are open problems.

(9) Finally, we do not know if any of the three problems FIREFIGHTER, DUAL FIREFIGHTER, and BOUNDED FIREFIGHTER are solvable in parameterized subexponential time on trees with respect to the standard parameters.

(10) An interesting research direction would be to complete the parameterized complexity landscape of the FIREFIGHTER problem with respect to structural graph parameters (see Figure 5.1).

Chapter

6 Conclusion

Motivated by the application of information diffusion through social networks, we analyzed the complexity of optimization problems involving a propagation process in a graph. Besides of being problems of practical relevance (social network analysis, epidemiology, ...), it seems to us that they are also of considerable theoretical interest.

Firstly, Target Set Selection generalizes well-known graph problems such as Dominating Set with thresholds [63], Vector Dominating Set [94], k-Tuple Dominating Set [76], (all these variants allow for only one "activation step"), Vertex Cover [35] (where the threshold value equals the vertex degree), Irreversible k-Conversion Set [51], r-Neighbor Bootstrap Percolation [14] (where the threshold of each vertex is k or r, respectively), and so-called dynamic monopolies [93] (where the threshold of a vertex v with degree $\deg(v)$ equals $\lceil \deg(v)/2 \rceil$). Furthermore, during our research we found out the existence of a close connection between the complementary optimization problem Max Open Influence and the Densest k-Subgraph problem. More precisely, we show that Max Open Influence with unanimity thresholds is at least as hard to approximate as Densest k-Subgraph. Due to this connection, we were able to provide an fpt-time approximation algorithm within any strictly increasing ratio with respect to the parameter k (independently established in [23]). Notably any improvement of the approximation ratio for Max Open Influence would automatically improve the ratio for Densest k-Subgraph. Recall that improving the approximation ratio of Densest k-Subgraph to a constant is a long open standing question. This makes Max Open Influence a problem of particular interest.

Secondly, it turns out that the previous two problems are also known as *cardinality constrained problems* [25]. Recall that a problem of this kind asks for finding a solution of k elements that optimizes an objective function. Several classical optimization problems admit a constrained version. For instance, Densest k-Subgraph is the constrained version of the Max Clique problem. From a classical complexity point of view, it is clear that an optimization problem shares the same complexity nature as its constrained version. However, from a parameterized and approximation perspective, this statement does not hold anymore. For example, the cardinality constrained version of Min Vertex Cover, denoted by Max k-Vertex Cover, asks to find a subset of at most k vertices that covers the maximum number of edges. The decision version has been proved W[1]-hard with respect to k [25] while Vertex Cover is well known to be fixed-parameter tractable. Notably, Marx [83] established an $(1+\varepsilon)$-approximation algorithm for Max k-Vertex Cover with running time $f(k,\varepsilon) \cdot n^{O(1)}$ for all $\varepsilon > 0$. As a consequence, it seems natural and interesting to ask the existence of an approximation algorithm for a cardinality constrained problem with running time $f(k) \cdot n^{O(1)}$ where k is the number of elements in the solution. However, it comes to us that the study of parameterized approximation of these kind of problems, in a general perspective, is a largely unexplored area. We left this as an interesting research direction.

Finally, it is worth noting that combining the parameterized complexity with approximation, in the context of cardinality constrained problems, can be done in a straightforward way as the number of elements in the solution is a natural parameter candidate. However, the story is not as simple for classical optimization problems since it is not clear

what could be the "standard" parameter. One might be tempted to choose the optimum value but this does not seem relevant since it is presumably hard to find it. To overcome this difficulty, the idea is to extend the input with an integer $k > 0$ (the value that an optimum should reach) which will play the role of the standard parameter. As a matter of fact, there exists several definitions of parameterized approximability based on that. In what follows, we will not review all of them and we refer the reader to the survey of Marx [83] for that purpose. Though we will discuss some possible extensions of our work toward that direction. Following the definition of [38], the Min Target Set Selection problem is *parameterized α-approximable* if the following holds: There exists a function α and an algorithm that, given any instance $I = (G, \text{thr})$ of Min Target Set Selection and an integer $k > 0$, computes in $f(k) \cdot |I|^{O(1)}$ time a target set S such that $|S| \leq k \cdot \alpha(k)$ if a target set of size at most k exists; an arbitrary output otherwise. We left open the question whether there exists such algorithm. Up to our knowledge there are not many positive or negative results in the literature for other optimization problems. Marx and Razgon [85] provided a parameterized 2-approximation algorithm for Edge Multicut. However, this last problem has been proved fixed-parameter tractable [24]. Nevertheless, Marx and Razgon introduced a first W[1]-hard problem which is parameterized constant-approximable. As pointed out in [84, 83], the existing negative results in the literature [50, 38] essentially rely on the "non monotone" nature of the corresponding problems. A minimization (resp. maximization) problem is said monotone if every superset (resp. subset) of a solution set is still a feasible solution. It follows that if a problem is non monotone then it may happen that all feasible solutions have the same size, thus making the search of approximate solutions as hard as the search of optimal ones. However, Min Target Set Selection is clearly monotone and, as mentioned above, generalizes several classical optimization problems. This makes its parameterized approximability study an other interesting research direction.

Appendix

A Compendium of problems

We give here the list of problems that are used in this document.

CLIQUE
Input: A graph $G = (V, E)$ and a positive integer k.
Question: Is there a clique $C \subseteq V$ of size at least k?

CUBIC MONOTONE 1-IN-3-SAT
Input: A CNF formula in which every clause contains exactly and only three positive literals and every variable appears in exactly three clauses.
Question: Is there a truth assignment to the variables such that each clause has exactly one true literal (such assignment is called a satisfying assignment)?

DENSEST k-SUBGRAPH
Input: A graph $G = (V, E)$ and a positive integer k.
Output: A subset $S \subseteq V$ of cardinality k that induces a maximum number of edges.

DOMINATING SET
Input: A graph $G = (V, E)$ and an integer k.
Question: Is there a subset $S \subseteq V$, $|S| \leq k$, such that for every vertex $v \in V \setminus S$ there is a vertex $u \in S$ with $uv \in E$?

HITTING SET
Input: A collection $\mathcal{F}$ of subsets of a finite set U and an integer $k \geq 0$.
Question: Is there a subset $U' \subseteq U$ with $U' \leq k$ such that U' contains at least one element from each subset in $\mathcal{F}$?

MIN COST FLOW
Input: A digraph $G = (V, A)$, a capacity function $\alpha : A \to \mathbb{N}$, a cost function $\beta : A \to \mathbb{Q}$ and a supply function $\gamma : V \to \mathbb{Q}$. If $\gamma(v) > 0$ then $\gamma(v)$ is a supply and if $\gamma(v) < 0$ then $\gamma(v)$ is a demand.
Output: A feasible flow of minimum cost *i.e.* a function $f : A \to \mathbb{N}$ that minimizes $\sum_{uv \in A} f(uv) \cdot \beta(uv)$ such that $\sum_{uv \in A} f(uv) - \sum_{vu \in A} f(vu) \geq \gamma(u)$ for all $u \in V$ and $f(uv) \leq \alpha(uv)$ for all $uv \in A$.

MIN VERTEX COVER
Input: A graph $G = (V, E)$.
Output: Find a subset $S \subseteq V$ of minimum size such that for all $uv \in E$ we have $u \in S$ or $v \in S$.

MAX INDEPENDENT SET
Input: A graph $G = (V, E)$.
Output: Find a subset $S \subseteq V$ of maximum size such that for every $u, v \in S$ we have $uv \in E$.

MAX CLIQUE
Input: A graph $G = (V, E)$.
Output: Find a clique $C \subseteq V$ of maximum size.

MAX E2SAT-3
Input: A CNF formula in which every clause contains exactly two literals and every variable appears in exactly three clauses.
Output: Find an assignment to the variables satisfying a maximum number of clauses.

MULTICOLORED CLIQUE
Input: A graph $G = (V, E)$, an integer k, and a coloring $\mathrm{col} : V \to \{1, \ldots, k\}$.
Question: Does G contain a multicolored clique of size k, that is, a vertex subset $V' \subseteq V$ with $|V'| = k$ such that for all $u, v \in V'$ it holds that $uv \in E$ and $\mathrm{col}(u) \neq \mathrm{col}(v)$?

RED/BLUE DOMINATING SET
Input: A bipartite graph $G = (R \cup B, E)$ and a positive integer k.
Question: Is there a set $R' \subseteq R$ of cardinality k such that every vertex in B has at least one neighbor in R'?

VERTEX COVER
Input: A graph $G = (V, E)$ and an integer k.
Question: Is there a subset $S \subseteq V$, $|S| \leq k$, such that for all $uv \in E$ we have $u \in S$ or $v \in S$?

Bibliography

[1] Ashkan Aazami and Kael Stilp. Approximation algorithms and hardness for domination with propagation. *SIAM Journal on Discrete Mathematics*, 23(3):1382–1399, 2009.

[2] Eyal Ackerman, Oren Ben-Zwi, and Guy Wolfovitz. Combinatorial model and bounds for target set selection. *Theoretical Computer Science*, 411(44-46):4017–4022, 2010.

[3] Sarah Spence Adams, Denise Sakai Troxell, and S. Luke Zinnen. Dynamic monopolies and feedback vertex sets in hexagonal grids. *Computers & Mathematics with Applications*, 62(11):4049–4057, 2011.

[4] Sarah Spence Adams, Paul Booth, Denise Sakai Troxell, and S. Luke Zinnen. Modeling the spread of fault in majority-based network systems: Dynamic monopolies in triangular grids. *Discrete Applied Mathematics*, 160(10-11):1624–1633, 2012.

[5] Ravindra Ahuja, Magnanti Thomas, and Orlin James. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[6] Paola Alimonti and Viggo Kann. Some APX-completeness results for cubic graphs. *Theoretical Computer Science*, 237(1-2):123–134, 2000.

[7] Elliot Anshelevich, Deeparnab Chakrabarty, Ameya Hate, and Chaitanya Swamy. Approximation algorithms for the firefighter problem: Cuts over time and submodularity. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC '09)*, LNCS 5878, pages 974–983. 2009.

[8] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007.

[9] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987.

[10] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[11] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 1st edition, 2009.

[12] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[13] Brenda S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[14] József Balogh, Béla Bollobás, and Robert Morris. Bootstrap percolation in high dimensions. *Combinatorics, Probability & Computing*, 19(5-6):643–692, 2010.

[15] Cristina Bazgan, Morgan Chopin, André Nichterlein, and Florian Sikora. Parameterized approximability of influence in social networks. In *Proceedings of the 19th Annual International Computing and Combinatorics Conference (COCOON '13)*, LNCS 7936, pages 543–554. 2013.

[16] Peter S. Bearman, James Moody, and Katherine Stovel. Chains of affection: The structure of adolescent romantic and sexual networks. *American Journal of Sociology*, 110(1):44–91, 2004.

[17] Oren Ben-Zwi, Danny Hermelin, Daniel Lokshtanov, and Ilan Newman. Treewidth governs the complexity of target set selection. *Discrete Optimization*, 8(1):87–96, 2011.

[18] Hans L. Bodlaender. Classes of graphs with bounded tree-width. *Bulletin of EATCS*, (116-128), 1988.

[19] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[20] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009.

[21] Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Proceedings of the 28th International Symposium on Theoretical Aspects of Computer Science (STACS '11)*, LIPIcs 9, pages 165–176. 2011.

[22] Hans Bodlaender L. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[23] Nicolas Bourgeois, Aristotelis Giannakos, Giorgio Lucarelli, Ioannis Milis, and Vangelis Th. Paschos. Exact and approximation algorithms for densest k-subgraph. In *Proceedings of the 7th International Workshop on Algorithms and Computation (WALCOM '13)*, LNCS 7748. 2013.

[24] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC '11)*, pages 459–468, 2011.

[25] Leizhen Cai. Parameterized complexity of cardinality constrained optimization problems. *The Computer Journal*, 51(1):102–121, 2008.

[26] Leizhen Cai, Siu Man Chan, and Siu On Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC '06)*, LNCS 4169, pages 239–250. 2006.

[27] Leizhen Cai, Elad Verbin, and Lin Yang. Firefighting on trees: (1 - 1/e)-approximation, fixed parameter tractability and a subexponential algorithm. In *Proceedings of the 19th International Symposium on Algorithms and Computation (ISAAC '08)*, LNCS 5369, pages 258–269. 2008.

[28] Carmen C. Centeno, Mitre C. Dourado, Lucia Draque Penso, Dieter Rautenbach, and Jayme L. Szwarcfiter. Irreversible conversion of graphs. *Theoretical Computer Science*, 412(29):3693 –3700, 2011.

[29] Damon Centola and Michael Macy. Complex contagions and the weakness of long ties. *American Journal of Sociology*, 113(3):702–734, 2007.

[30] Marco Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67(4):654–685, 2003.

[31] Parinya Chalermsook and Julia Chuzhoy. Resource minimization for fire containment. In *Proceedings of the 21th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '10)*, pages 1334–1349, 2010.

[32] Ching-Lueh Chang and Yuh-Dauh Lyuu. Spreading messages. *Theoretical Computer Science*, 410 (27–29):2714–2724, 2009.

[33] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.

[34] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40–42):3736–3756, 2010.

[35] Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[36] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*, pages 199–208, 2009.

[37] Wei Chen, Chi Wang, and Yajun Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '10)*, pages 1029–1038, 2010.

[38] Y Chen, M Grohe, and M Grüber. On parameterized approximability. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC '06)*, LNCS 4169, pages 109–120. 2006.

[39] Chung-Ying Chiang, Liang-Hao Huang, Bo-Jr Li, Jiaojiao Wu, and Hong-Gwa Yeh. Some results on the target set selection problem. *Journal of Combinatorial Optimization*, 2012. Online available.

[40] Nicholas A. Christakis and James H. Fowler. The Spread of Obesity in a Large Social Network over 32 Years. *New England Journal of Medicine*, 357(4):370–379, 2007.

[41] Claire Christensen, István Albert, Bryan Grenfell, and Réka Albert. Disease dynamics in a dynamic social network. *Physica A: Statistical Mechanics and its Applications*, 389(13):2663–2674, 2010.

[42] Fan R.K. Chung and Paul D. Seymour. Graphs with small bandwidth and cutwidth. In *Graph Theory and combinatorics 1988 Proceedings of the Cambridge Combinatorial Conference in Honour of Paul Erdös*, volume 43 of *Annals of Discrete Mathematics*, pages 113–119. 1989.

[43] Ernest J. Cockayne, R. M. Dawes, and Stephen T. Hedetniemi. Total domination in graphs. *Networks*, 10(3):211–219, 1980.

[44] Bruno Courcelle. The monadic second-order logic of graphs I. Recognizable sets of Finite Graphs. *Information and Computation*, pages 12–75, 1990.

[45] Mike Develin and Stephen G. Hartke. Fire containment in grids of dimension three and higher. *Discrete Applied Mathematics*, 155(17):2257–2268, 2007.

[46] Irit Dinur and Shmuel Safra. The importance of being biased. In *Proceedings of the 34th annual ACM symposium on Theory of computing (STOC '02)*, pages 33–42, 2002.

[47] Pedro Domingos and Matthew Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '01)*, pages 57–66, 2001.

[48] Martin Doucha and Jan Kratochvíl. Cluster vertex deletion: A parameterization between vertex cover and clique-width. In *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS '12)*, LNCS 7464, pages 348–359. 2012.

[49] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999.

[50] Rodney G. Downey, Michael R. Fellows, Catherine McCartin, and Frances Rosamond. Parameterized approximation of dominating set problems. *Information Processing Letters*, 109(1):68–70, 2008.

[51] Paul A. Dreyer and Fred S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

[52] David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010.

[53] David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.

[54] Stephen Finbow and Gary MacGillivray. The firefighter problem: a survey of results, directions and questions. *The Australasian Journal of Combinatorics*, 43:57–77, 2009.

[55] Stephen Finbow, Andrew King, Gary MacGillivray, and Romeo Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16):2094–2105, 2007.

[56] Peter Floderus, Andrzej Lingas, and Mia Persson. Towards more efficient infection and fire fighting. In *Proceedings of the 18th Computing: the Australasian Theory Symposium (CATS '11)*, CRPIT 119, pages 69–74. 2011.

[57] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[58] Fedor Fomin, Pinar Heggernes, and Erik Jan van Leeuwen. Making life easier for firefighters. In *Proceedings of the 6th International conference on Fun with Algorithms (FUN '12)*, LNCS 7288, pages 177–188. 2012.

[59] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, pages 133–142, 2008.

[60] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[61] Petr A. Golovach, Jan Kratochvil, and Ondřej Suchý. Parameterized complexity of generalized domination problems. *Discrete Applied Mathematics*, 160(6):780–792, 2012.

[62] Rudolf Halin. s-functions for graphs. *Journal of Geometry*, 8(1-2):171–186, 1976.

[63] Jochen Harant, Anja Pruchnewski, and Margit Voigt. On dominating sets and independent sets of graphs. *Combinatorics, Probability and Computing*, 8(6):547–553, 1999.

[64] Bert Hartnell. Firefighter! an application of domination, Presentation. In *10th Conference on Numerical Mathematics and Computing, University of Manitoba in Winnipeg, Canada*, 1995.

[65] Bert Hartnell and Qiyan Li. Firefighting on trees: how bad is the greedy algorithm? *Congressus Numerantium*, 145:187–192, 2000.

[66] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[67] Dorit S. Hochbaum. *Approximation algorithms for NP-hard problems*. PWS Publishing Co., 1997.

[68] Russell Impagliazzo and Ramamohan Paturi. The Complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[69] Yutaka Iwaikawa, Naoyuki Kamiyama, and Tomomi Matsui. Improved approximation algorithms for firefighter problem on trees. *IEICE Transactions on Information ans Systems*, E94.D(2):196–199, 2011.

[70] Qingye Jiang, Guojie Song, Gao Cong, Yu Wang, Wenjun Si, and Kunqing Xie. Simulated annealing based influence maximization in social networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI '11)*, 2011.

[71] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[72] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, pages 137–146, 2003.

[73] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh V. Vazirani. On syntactic versus computational views of approximability. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS '94)*, pages 819–830, 1994.

[74] Masahiro Kimura, Kazumi Saito, and Hiroshi Motoda. Blocking links to minimize contamination spread in a social network. *ACM Transactions on Knowledge Discovery from Data*, 3(2):9:1–9:23, 2009.

[75] Andrew King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

[76] Ralf Klasing and Christian Laforest. Hardness results and approximation algorithms of k-tuple domination in graphs. *Information Processing Letters*, 89(2):75–83, 2004.

[77] Christian Komusiewicz and Rolf Niedermeier. New races in parameterized algorithmics. In *Proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS '12)*, LNCS 7464, pages 19–30. 2012.

[78] Ephraim Korach and Nir Solel. Tree-width, path-width, and cutwidth. *Discrete Applied Mathematics*, 43(1):97–101, 1993.

[79] Hendrik W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[80] Jure Leskovec, Lada A. Adamic, and Bernardo A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web*, 1(1), 2007.

[81] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '07)*, pages 420–429, 2007.

[82] Gary MacGillivray and Ping Wang. On the firefighter problem. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 47:83–96, 2003.

[83] Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51 (1):60–78, 2008.

[84] Dániel Marx. Completely inapproximable monotone and antimonotone parameterized problems. *Journal of Computer and System Sciences*, 79(1):144–151, 2013.

[85] Dániel Marx and Igor Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Information Processing Letters*, 109(20):1161–1166, 2009.

[86] Luke Mathieson and Stefan Szeider. Parameterized graph editing with chosen vertex degrees. In *Proceedings of the 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA '08)*, LNCS 5165, pages 13–22. 2008.

[87] Ross M. McConnell and Jeremy Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '94)*, pages 536–545, 1994.

[88] Cristopher Moore and John M. Robson. Hard tiling problems with simple tiles. *Discrete & Computational Geometry*, 26(4):573–590, 2001.

[89] Kah Loon Ng and Paul Raff. A generalization of the firefighter problem on ZxZ. *Discrete Applied Mathematics*, 156(5):730–745, 2008.

[90] André Nichterlein, Rolf Niedermeier, Johannes Uhlmann, and Mathias Weller. On tractable cases of target set selection. *Social Network Analysis and Mining*, 2012. Online available.

[91] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[92] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991.

[93] David Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theoretical Computer Science*, 282(2):231–257, 2002.

[94] Venkatesh Raman, Saket Saurabh, and Sriganesh Srihari. Parameterized algorithms for generalized domination. In *Proceedings of the 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA'08)*, LNCS 5165, pages 116–126. 2008.

[95] T. V. Thirumala Reddy and C. Pandu Rangan. Variants of spreading messages. *Journal of Graph Algorithms and Applications*, 15(5):683–699, 2011.

[96] Daniel Reichman. New bounds for contagious sets. *Discrete Mathematics*, 312(10):1812–1814, 2012.

[97] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pages 61–70, 2002.

[98] Neil Robertson and Paul D. Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[99] Allan E. Scott, Ulrike Stege, and Norbert Zeh. Politician's firefighting. In *Proceedings of the 17th International Symposium on Algorithms and Computation (ISAAC '06)*, LNCS 4288, pages 608–617. 2006.

[100] Jan Arne Telle. Complexity of domination-type problems in graphs. *Nordic Journal of Computing*, 1(1):157–171, 1994.

[101] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[102] Ping Wang and Stephanie A. Moeller. Fire control on graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 41:19–34, 2002.

[103] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[104] Lin Yang. Efficient algorithms on trees. M. Phil thesis, Department of Computer Science and Engineering, The Chinese University of Hong Kong, 2009.

Index

Résumé. Dans cette thèse, nous étudions la complexité algorithmique de problèmes d'optimisation impliquant un processus de diffusion dans un graphe. Plus précisément, nous nous intéressons tout d'abord au problème de sélection d'un ensemble cible. Ce problème consiste à trouver le plus petit ensemble de sommets d'un graphe à "activer" au départ tel que tous les autres sommets soient activés après un nombre fini d'étapes de propagation. Si nous modifions ce processus en permettant de "protéger" un sommet à chaque étape, nous obtenons le problème du pompier dont le but est de minimiser le nombre total de sommets activés en protégeant certains sommets. Dans ce travail, nous introduisons et étudions une version généralisée de ce problème dans laquelle plus d'un sommet peut être protégé à chaque étape. Nous proposons plusieurs résultats de complexité pour ces problèmes à la fois du point de vue de l'approximation mais également de la complexité paramétrée selon des paramètres standards ainsi que des paramètres liés à la structure du graphe.

Mots clefs: Optimisation combinatoire, théorie des graphes, algorithmes d'approximation, complexité paramétrée, approximation paramétrée, réseaux sociaux, problème du pompier, selection d'un ensemble cible.

Abstract. In this thesis, we investigate the computational complexity of optimization problems involving a "diffusion process" in a graph. More specifically, we are first interested to the target set selection problem. This problem consists of finding the smallest set of initially "activated" vertices of a graph such that all the other vertices become activated after a finite number of propagation steps. If we modify this process by allowing the possibility of "protecting" a vertex at each step, we end up with the firefighter problem that asks for minimizing the total number of activated vertices by protecting some particular vertices. In fact, we introduce and study a generalized version of this problem where more than one vertex can be protected at each step. We propose several complexity results for these problems from an approximation point of view and a parameterized complexity perspective according to standard parameterizations as well as parameters related to the graph structure.

Keywords: Combinatorial optimization, graph theory, approximation algorithms, parameterized complexity, parameterized approximation, social networks, firefighter problem, target set selection.