Hardness results and approximation algorithms for some problems on graphs

by

Ashkan Aazami

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2008

Abstract

This thesis has two parts. In the first part, we study some graph covering problems with a non-local covering rule that allows a "remote" node to be covered by repeatedly applying the covering rule. In the second part, we provide some results on the packing of Steiner trees.

In the Propagation problem we are given a graph G and the goal is to find a minimum-sized set of nodes S that covers all of the nodes, where a node v is covered if (1) v is in S, or (2) v has a neighbor u such that u and all of its neighbors except v are covered. Rule (2) is called the *propagation* rule, and it is applied iteratively. Throughout, we use n to denote the number of nodes in the input graph. We prove that the path-width parameter is a lower bound for the optimal value. We show that the Propagation problem is NP-hard in planar weighted graphs. We prove that it is NP-hard to approximate the optimal value to within a factor of $2^{\log^{1-\epsilon} n}$ in weighted (general) graphs.

The second problem that we study is the Power Dominating Set problem. This problem has two covering rules. The first rule is the same as the *domination* rule as in the Dominating Set problem, and the second rule is the same propagation rule as in the Propagation problem. We show that it is hard to approximate the optimal value to within a factor of $2^{\log^{1-\epsilon} n}$ in general graphs. We design and analyze an approximation algorithm with a performance guarantee of $O(\sqrt{n})$ on planar graphs.

We formulate a common generalization of the above two problems called the General Propagation problem. We reformulate this general problem as an orientation problem, and based on this reformulation we design a dynamic programming algorithm. The algorithm runs in linear time when the graph has tree-width $O(1)$. Motivated by applications, we introduce a restricted version of the problem that we call the ℓ-round General Propagation problem. We give a PTAS for the ℓ-round General Propagation problem on planar graphs, for small values of ℓ. Our dynamic programming algorithms and the PTAS can be extended to other problems in networks with similar propagation rules. As an example we discuss the extension of our results to the Target Set Selection problem in the threshold model of the diffusion processes.

In the second part of the thesis, we focus on the Steiner Tree Packing problem. In this problem, we are given a graph G and a subset of terminal nodes $R \subseteq V(G)$. The goal in this problem is to find a maximum cardinality set of disjoint trees that each spans R, that is, each of the trees should contain all terminal nodes. In the *edge-disjoint* version of this problem, the trees have to be edge disjoint. In the *element-disjoint* version, the trees have to be node disjoint on non-terminal nodes and edge-disjoint on edges adjacent to terminals. We show that both problems are NP-hard when there are only 3 terminals. Our main focus is on planar instances of these problems. We show that the edge-disjoint version of the problem is NP-hard even in planar graphs with 3 terminals on the same face of the embedding.

Next, we design an algorithm that achieves an approximation guarantee of $\frac{1}{2} - \frac{1}{k}$, given a planar graph that is k element-connected on the terminals; in fact, given such a graph the algorithm returns $k/2 - 1$ element-disjoint Steiner trees. Using this algorithm we get an approximation algorithm with guarantee of (almost) 4 for the edge-disjoint version of the problem in planar graphs. We also show that the natural LP relaxation of the edge-disjoint Steiner Tree Packing problem has an integrality ratio of $2 - \epsilon$ in planar graphs.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Joseph Cheriyan, who has supported me with his encouragement, guidance and patience. I thank Joseph for his insightful suggestions and comments on my research, talks and especially this thesis. Also, I would like to thank the faculty and the staff of the Combinatorics and Optimization Department for providing such an excellent academic environment.

Special thanks to my collaborators, Krishnam Raju Jampani, and Michael Stilp. Also, I would like to thank my thesis committee, Shai Ben-David, Jochen Könemann, David Shmoys, and Nick Wormald, for taking the time and effort to carefully read my thesis and attend my defense.

I am indebted to my parents, Farhad Aazami and Maliheh Sobhi, for their love and their support during these years. Last but not least, I am very grateful to my wife, Sara; without her love, support and sacrifices this thesis would have been impossible. This thesis is dedicated to her.

Contents

List of Figures

Chapter 1

Introduction

This thesis is divided into two parts. In the first part, we study some graph covering problems with a non-local covering rule that allows a "remote" node to be covered by repeatedly applying the covering rule. The PROPAGATION and POWER DOMINATING SET problems are key problems in this setting. We provide approximation algorithms and hardness results for both problems. In the second part, we study the STEINER TREE PACKING problem. Our main focus is on planar graphs, and we provide approximation algorithms and hardness results for this problem.

Some of the results in Chapter 2 are based on discussions with Simone Severini, Sang-il Oum, and Jim Geelen. The results in Chapter 3 and some of the results in Chapter 4 are based on joint work with Michael Stilp that appeared in [3, 4]. Some of the results in Chapter 4 have appeared in [1]. The results in Chapter 5 are based on joint work with K. Raju Jampani and my supervisor [2].

1.1 Overview of results

In this section, an overview of the results presented in this thesis is discussed. The previous literature and applications are discussed in detail in the relevant chapters.

1.1.1 Propagation problems

In the PROPAGATION problem, we are given a graph G and the goal is to find a minimum-sized set of nodes S that covers all of the nodes, where a node v is covered if (1) v is in S, or (2) v has a neighbor u such that u and all of its neighbors except v are covered. Rule (2) is called the *propagation* rule, and it is applied iteratively. Throughout, we use n to denote the number of nodes in the input graph. We provide two lower bounding techniques for this problem. The first one is based on the notion of *strong regions*. The number of disjoint strong regions provides a lower bound for the optimal value. The second lower bound that we provide is based on the path-width parameter from graph minor theory. We show that the path-width is a also a lower bound for the optimal value. We show that the Propagation problem is NP-hard in planar weighted graphs. In general graphs, we prove a strong hardness of approximation result, which shows that the weighted problem cannot have a polynomial-time approximation algorithm with poly-logarithmic guarantee.

The second problem that we study is the Power Dominating Set (abbreviated as PDS) problem. This problem has two covering rules. The first rule is the same as the *domination* rule as in the Dominating Set problem; the *domination* rule says that a node v is covered if it is picked or one of its neighbors is picked. The second rule is the same propagation rule as in the Propagation problem. Guo, Niedermeier, and Raible [30] initiated comparisons between the complexity of the Dominating Set problem and the PDS problem on special classes of graphs. We show that it is NP-hard to approximate the optimal value of the PDS problem to within a poly-logarithmic factor in unweighted graphs. This seems to be the first known "separation" result between the two problems. We introduce the notion of *strong regions* as a means of obtaining lower bounds on the optimal value. Based on this, we develop an approximation algorithm with guarantee of $(k+1)$ for graphs that have tree-width k. This gives an approximation algorithm with guarantee of $O(\sqrt{n})$ on planar graphs. Moreover, we show that this is the best guarantee that we can get using our lower bound. We extend the PDS problem to directed graphs, and we show that even in *directed acyclic* graphs, PDS is hard to approximate within the same threshold as for undirected graphs. We also introduce another extension of the PDS problem by introducing a parameter that restricts the number of "parallel" applications of the propagation rule. We show that even allowing four rounds of parallel propagation makes the problem as hard as PDS.

We formulate a common generalization of the above two problems called the General Propagation problem. We reformulate this general problem as an orientation problem, and based on this reformulation we design a dynamic programming algorithm to optimally solve the general problem. The algorithm runs in linear time when the graph has tree-width $O(1)$. Motivated by applications, we introduce a restricted version of the problem called the ℓ-round General Propagation problem. We also reformulate this problem as an orientation problem, and based on this reformulation we design a dynamic programming algorithm for the restricted problem. We extend Baker's [7] polynomial time approximation scheme (PTAS) to obtain a PTAS for the ℓ-round General Propagation problem on planar graphs, for small values of ℓ. This algorithm needs to solve instances of the ℓ-round General Propagation problem on graphs that have bounded tree-width. Our dynamic programming algorithms and the PTAS for the General Propagation problem can be extended to other problems in networks with similar propagation rules. As an example we discuss the extension of our results to the Target Set Selection problem in the threshold model of the diffusion process.

1.1.2 Steiner tree packing problems

In the second part of the thesis, we focus on the Steiner Tree Packing problem and study approximation algorithms and hardness results for planar instances of the problem. In this problem, we are given a graph G and a subset of terminal nodes $R \subseteq V(G)$. The goal in this problem is to find a maximum cardinality set of disjoint trees that each spans R; that is, each of the trees should contain all terminal nodes. We study two versions of the problem. In the *edge-disjoint* version, the Steiner trees are required to be edge-disjoint; that is, each edge of the graph should be in at most one Steiner tree. In the *element-disjoint* version, the Steiner trees are required to be element-disjoint; that is, each element (i.e., edge or non-terminal node) should be in at most one tree. We show that the element-disjoint version of the problem in general graphs is NP-hard when there are only 3 terminals. We show that the edge-disjoint version of the problem is NP-hard even in planar graphs with 3 terminals on the same face of the embedding. We prove that if we have a planar graph such that the terminal nodes are k element-connected, then there are at least $k/2 - 1$ element-disjoint Steiner trees. This provides an approximation algorithm with guarantee of (almost) 2 for the element-disjoint version of the problem on planar graphs. Based on this algorithm, we get an approximation algorithm with guarantee of (almost) 4 for the edge-disjoint version of the problem on planar graphs. We show that the standard linear programming relaxation of the edge-disjoint Steiner Tree Packing problem has an integrality ratio of at least $2 - o(1)$ in planar graphs. Moreover, we modify our construction to get a similar result for the element-disjoint version of the problem.

1.2 Presentation overview

In Chapter 2 we present our lower bounds and hardness results on the Propagation problem.

In Chapter 3 we present our approximation algorithm and hardness result on the PDS problem. We also introduce two extensions of the PDS problem and prove hardness results for them.

In Chapter 4 we introduce a generalization of the Propagation and PDS problems and present a reformulation for this generalized problem in terms of an orientation problem. We present a PTAS for the ℓ-round version of this problem on planar graphs for small values of the parameter ℓ.

In Chapter 5, we study the Steiner Tree Packing problem. Our focus is on planar graphs, and we present approximation algorithms and hardness results for the edge-disjoint and element-disjoint versions of the problem.

1.3 Preliminary definitions and notations

Let $G = (V, E)$ be an undirected graphs. A node u is called a *neighbor* of v if there is an edge between u and v in G, i.e., $\{u, v\} \in E$. The *open neighborhood* of a node v, denoted by $N(v)$, is the set of all neighbors of a node v. The *closed neighborhood* of a node v, denoted by $N[v]$, is defined as $\{v\} \cup N(v)$. The degree of a node v is denoted by $d_G(v)$. An *x-y path* in G is a sequence $u_0 = x, u_1, \ldots, u_{k-1}, u_k = y$ of nodes such that there is an edge in G between u_i and u_{i+1}, for $i = 0, \ldots, k-1$.

Let $G = (V, E)$ be a directed graph. A node w is called an *out-neighbor* of a node v if there is a directed edge from v to w in G. Similarly, w is called an *in-neighbor* of v if the directed edge (w, v) is present. The number of out-neighbors of v is called the *out-degree* of v and is denoted by $d^+_G(v)$, the in-degree $d^-_G(v)$ is defined similarly. The directed graphs that we consider have no loops nor parallel edges, but may have two edges with different directions on the same two end nodes (we call such edges *antiparallel*). Given a directed graph G, by the *underlying undirected graph* we mean the undirected graph obtained from G by removing the direction of edges and also removing any parallel edges that are introduced.

The graphs that we consider may have weights assigned to their nodes. We denote the weight function by $\mathtt{W}_G : V \to \mathbb{R}$. The weight function $\mathtt{W}$ should not be confused with the symbol w which may be used to denote a node. Given a subset S of nodes we use the notation $\mathtt{W}_G(S)$ to denote the sum of the weights of the nodes in S; i.e., $\mathtt{W}_G(S) = \sum_{v \in S} \mathtt{W}_G(v)$.

We may drop the subscript G in our notation when the graph G is clear from the context.

For notation and terminology in graph theory, approximation algorithms, and computational complexity we follow Diestel's book on graph theory [23], Vazirani's book on approximation algorithms [72], and Garey and Johnson's book [28].

Part I

Propagation problems

Chapter 2

The propagation problem

In the Propagation problem, we are given an undirected graph $G = (V, E)$ and the goal is to pick a minimum-size set S of nodes which can "cover" all nodes of G. The set of nodes that can be covered by S, denoted by $\mathcal{P}^*(S)$, is defined in the following way.

(R1) Start with $\mathcal{P}(S) = S$ and sequentially apply rule (R2),

(R2) (propagation) If a node u is in $\mathcal{P}(S)$, one of its neighbors v is not in $\mathcal{P}(S)$, and all other neighbors of u are in $\mathcal{P}(S)$, then we add v to $\mathcal{P}(S)$.

Rule (R2) is called the propagation rule. The set $\mathcal{P}(S)$ changes as we repeatedly apply the propagation rule, until every node in $\mathcal{P}(S)$ either has zero neighbors in $V \setminus \mathcal{P}(S)$ or has at least two neighbors in $V \setminus \mathcal{P}(S)$. Let $\mathcal{P}^*(S)$ denote the final set $\mathcal{P}(S)$; see Proposition 2.1.1.

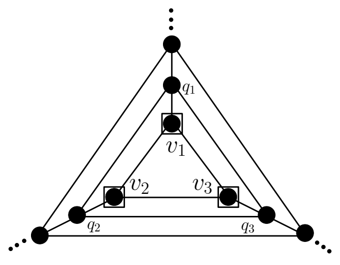


Figure 2.1: Example for the Propagation problem; the nodes enclosed in boxes are the picked nodes.

Consider the graph in Figure 2.1; the graph has t disjoint triangles, and three disjoint paths such that each has exactly one node from each triangle. We claim that the set $S = \{v_1, v_2, v_3\}$ consisting of the three nodes in the innermost triangle covers the whole graph. To see this, first note that the propagation rule applies to node v_1 since $\mathcal{P}(S) = S$ contains all neighbors of v_1 except the neighbor in the second triangle, q_1, so we add q_1 to $\mathcal{P}(S)$. Similarly, we apply the propagation rule to v_2 to add q_2 to $\mathcal{P}(S)$, and then we apply the propagation rule to v_3 to add q_3 to $\mathcal{P}(S)$. Thus, we added the nodes of the second triangle to $\mathcal{P}(S)$ by applying the propagation rule sequentially to the nodes of the first triangle. Next, we add the nodes of the third triangle to $\mathcal{P}(S)$ by applying the propagation rule sequentially to the nodes of the second triangle. By repeating this for each triangle, we eventually add all the nodes of G to $\mathcal{P}(S)$; thus, $\mathcal{P}^*(S) = V$.

Throughout this chapter, we focus on undirected graphs and we use G to denote the input graph for the PROPAGATION problem. We denote the size of an optimal solution for the PROPAGATION problem by $\rho(G)$. We say that a node v is *picked* if v is in the solution that we are considering. When a node v is inserted into $\mathcal{P}(S)$ by applying rule (R2) to node u, we say that v is *covered* by applying the propagation rule to u and we denote this by $u \to v$; note that all neighbors of u except v must be in $\mathcal{P}(S)$ before the propagation rule can apply. Given a weight function, $\mathtt{W} : V(G) \to \mathbb{R}$, defined on the nodes of G, we can also ask to find a set S with minimum weight that covers all nodes of G. We denote the weight of an optimal solution by $\rho(G, \mathtt{W})$, or simply by $\rho(G)$ when the weight function $\mathtt{W}$ is clear from the context. Thus, $\rho(G, \mathtt{W}) = \min_{S \subseteq V} \{\mathtt{W}(S) | \mathcal{P}^*(S) = V\}$.

The PROPAGATION problem is a fundamental problem that is closely related to a number of topics of current interest. We discuss connections to several areas: *power dominating sets*, *zero forcing sets*, *quantum networks*, and *influence in social networks*.

- The POWER DOMINATING SET (PDS) problem arose in the context of electric power networks, where the aim is to monitor all of the network by placing a minimum-size set of very expensive devices called phase measurement units; these units have the capability of monitoring remote elements via propagation (as in rule (R2)); see Brueni [13], Baldwin et al. [8], and Mili et al. [54]. In the engineering literature, the problem is called the PMU placement problem. The PDS problem has two covering rules. The first rule of PDS is the *domination* rule of the DOMINATING SET problem, and this rule is applied only once. In more detail, if $v \in S$ then we add v and all of its neighbors to $\mathcal{P}(S)$. The second rule of PDS is exactly the same as rule (R2) in the PROPAGATION problem, and this rule is applied sequentially. For more discussion on the PDS problem refer to Chapter 3.

- The authors of [29] formulated a problem called ZERO FORCING SET which is the same as the PROPAGATION problem. They formulated this problem in order to provide a lower bound for the minimum *rank* of a graph G, where the minimum rank of G is defined to be the smallest possible rank over all symmetric real matrices whose ijth entry is non-zero if and only if $\{i, j\}$ is an edge in G. Let $\mathbf{m}(G)$ denote this minimum rank. The authors showed that $|V(G)| - \rho(G) \leq \mathbf{m}(G)$. Using this inequality, they proved tight bounds on $\rho(G)$ for some classes of graphs such as the d-dimensional cube, the Cartesian product of two paths, etc. Independently of [29], Alon [6] proved tight bounds on $\rho(G)$ for d-dimensional cubes and some classes of *Cayley* graphs using similar ideas as in [29].

- Researchers in Quantum Information Processing have recently focused on theoretical issues in the control of quantum networks, for example, a network of quantum particles of spin one half. See [15], where the context is described from the perspective of Quantum Physics. The graph theoretical model for the propagation problem was introduced in [68], where some results on special classes of graphs have been presented.

- Social networks are a topic of intense study since they are useful in modeling many phenomena in sociology, economics, epidemiology and engineering. In particular, current applied and theoretical research has focused on diffusion processes (spread of influence) and their characteristics [40, 41, 56, 61, 16, 55, 62, 10, 64, 65, 52]. Research in theory has led to the introduction of models such as the *threshold model* [40, 41]. These models are closely related to our *propagation* model; although neither model is a special case of the other one, some of the fundamental issues underlying hardness of approximation are the same for both settings (see [4, 16]).

The main results in this chapter are as follows:

- In section 2.2, we present a lower bound on $\rho(G)$ via the path-width parameter.

- In Section 2.3, we focus on the complexity of computing $\rho(G)$ in different classes of graphs. We prove that the PROPAGATION problem is NP-hard in planar graphs with nodes of weight zero or one. Next, we prove that it is NP-hard to approximate $\rho(G)$ within a factor of $2^{\log^{1-\epsilon} n}$ in weighted undirected graphs. This hardness result rules out algorithms with poly-logarithmic approximation guarantees.

Moreover, in Chapter 4, we present a linear time dynamic programming algorithm to compute $\rho(G)$ for weighted graphs of bounded tree-width.

2.1 Preliminaries on Propagation

This section presents some simple and useful observations on the PROPAGATION problem.

Proposition 2.1.1 *For any graph $G = (V, E)$ and a set $S \subseteq V$, any sequence of applications of the propagation rule results in the same set $\mathcal{P}^*(S)$.*

Proof: Starting from $\mathcal{P}(S) = S$, consider two different sequences of application of the propagation rule, and let $\mathcal{P}^{'}(S)$ and $\mathcal{P}^{''}(S)$ denote the final set $\mathcal{P}(S)$ for the first and the second sequence, respectively. Suppose that $\mathcal{P}^{'}(S) \neq \mathcal{P}^{''}(S)$. Focus on an "earliest node" v that is in exactly one of $\mathcal{P}^{'}(S)$ or $\mathcal{P}^{''}(S)$; in other words, we order the nodes in $\mathcal{P}^{'}(S)$ and $\mathcal{P}^{''}(S)$ according to the order of covering, and take v to be the first node of either $\mathcal{P}^{'}(S)$ or $\mathcal{P}^{''}(S)$ that is in exactly one of $\mathcal{P}^{'}(S)$ or $\mathcal{P}^{''}(S)$. We get a contradiction since the nodes which come before v in one sequence are also in the other sequence; hence, if v can be covered by an application of the propagation rule for one sequence, then v can be covered in the other sequence too. □

Now, we reformulate the PROPAGATION problem as an orientation problem; for more discussion on this reformulation refer to Section 4.1. An *orientation* of an undirected graph $G = (V, E)$ is obtained by assigning an orientation to some (but possibly not all) edges of G. We denote an orientation of G by $\widehat{\mathcal{O}} = (V, E_d, E_u)$, where E_d is the set of oriented edges and E_u is the set of unoriented edges. Informally speaking, in our reformulation, the oriented edges denote the way the propagation rule applies.

Definition 2.1.2 *A* valid orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ *of an undirected graph* $G = (V, E)$ *is an orientation of* G *with the following properties:*

1. *Each vertex of the graph* $G_d = (V, E_d)$ *has in-degree and out-degree of at most* 1*;* $\forall v \in G : d^-_{G_d}(v), d^+_{G_d}(v) \leq 1$.
2. *G has no* dependency cycle. *A dependency cycle C is a cycle in G such that all oriented edges in C are in the same direction and it has no two consecutive unoriented edges.*

A node with in-degree 0 *in* $G_d = (V, E_d)$ *is called a* source *of* $\widehat{\mathcal{O}}$.

The following theorem is a special case of Theorem 4.1.4 that is proved in Section 4.1.

Theorem 2.1.3 *Any graph* $G = (V, E)$ *has a valid orientation with* S *as the set of sources if and only if* $\mathcal{P}^*(S) = V$.

Consider a valid orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ of a graph $G = (V, E)$. Observe that the oriented subgraph $G_d = (V, E_d)$ is a disjoint union of directed paths; note that there is a directed path corresponding to each source node in $\widehat{\mathcal{O}}$. Thus, by applying the above theorem to trees, we get the following corollary.

Corollary 2.1.4 *For any tree* T,

$$\rho(T) = \textit{minimum number of node-disjoint paths that covers } V(T)$$

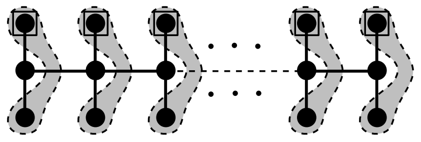


Figure 2.2: Caterpillar graph

We show that $\rho(G)$ is not a monotone function under edge deletion and edge contraction.

Proposition 2.1.5 *Removing edges in a graph G may increase or decrease $\rho(G)$.*

Proof: Consider the *wheel* graph, W_m, on $m+1$ nodes; W_m is obtained by adding a node (called the *center* node) to a cycle on m nodes and connecting the center node to all nodes on the cycle. If we pick the center node and two consecutive nodes from the cycle, then we can cover all nodes, so $\rho(W_m) \leq 3$. Now, delete all edges of the cycle to get a star, S_m, on $m+1$ nodes. It can be seen that $\rho(S_m) \geq m-1$. To see this, observe that if at least two leaf nodes are not picked, then the propagation rule fails to apply to the center node, so the leaf nodes that are not picked will not be covered. This shows that deleting edges may increase $\rho(G)$.

The complete graph, K_m, has $\rho(K_m) = m-1$. By removing some edges from K_m we can get the path, P_m, on m nodes which has $\rho(P_m) = 1$. Hence, deleting edges may decrease $\rho(G)$. $\square$

Proposition 2.1.6 *Contracting edges in a graph G may increase or decrease $\rho(G)$.*

Proof: Consider a caterpillar graph, L_m, on $3m$ nodes; L_m is obtained from a path P on m nodes by attaching two isolated nodes to each node on P (see Figure 2.2). Observe that $\rho(L_m) \leq m$ because we get a solution of size m by picking one of the two leaf neighbors of each node of P (the boxed nodes in Figure 2.2 form one such solution). By contracting all edges of the path P, we get a star S_{2m}, which has $\rho(S_{2m}) \geq 2m-1$. Thus, contracting edges may increase $\rho(G)$.

Now contract all leaf edges of L_m to get a path P_m on m nodes, which has $\rho(P_m) = 1$. Hence, contracting edges may decrease $\rho(G)$. $\square$

Let G' be the graph that is obtained by subdividing all edges of a caterpillar graph L_m. We claim that G' has $\rho(G') = 2m-1$. Thus, subdividing edges may increase $\rho(G)$. Now, we show that subdividing an edge never decreases $\rho(G)$.

Proposition 2.1.7 *Let G' be the graph that is obtained by subdividing an edge of a graph G. Then we have $\rho(G) \leq \rho(G')$.*

Proof: Let G' be obtained from G by subdividing an edge $e = \{u, v\}$, and let x be the subdividing node in G'. Assume that S' is an optimal solution for G'; that

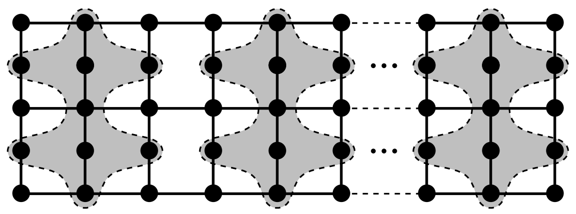


Figure 2.3: Modified $3 \times 3m$ grid

is, $\rho(G') = |S'|$ and $\mathcal{P}^*(S') = V(G')$. We claim that if $x \notin S'$, then $S = S'$ is a solution for G. Otherwise, removing x from S' and adding either u or v (picking the one which is covered later) to S' we get a solution for G. Proving this claim shows that $\rho(G) \leq |S| \leq |S'| = \rho(G')$. Consider the following two cases:

1. Suppose that $x \notin S'$. Assume that x is covered in G' by applying the propagation rule to u (i.e., $u \to x$). By picking S' all applications of the propagation rule occurring before $u \to x$ in G' can also occur in G. If v is currently not covered in G, then the propagation $u \to v$ occurs and v is covered. Next, all other propagations as in G' can occur in G. Thus, S' is a solution to G.

2. Assume that $x \in S'$, and u is covered before v in G'. Then $S = (S' \setminus \{x\}) \cup \{v\}$ is a solution to G. To see this, consider the sequence of propagations $x_i \to y_i$ that occur in G' before and including the step when u is covered; clearly, the same sequence of propagations can occur in G. Also, assuming that v is picked in G, we can "replicate" in G the sequence of propagations that occur in G' after u is covered.

This completes the proof, and shows that subdividing an edge can never decrease $\rho(G)$. □

Let $G = (V, E)$ be an unweighted graph. A set of nodes $R \subseteq V$ is called a *strong* region if any node not in R has either no neighbor in R or has at least two neighbors in R; that is, $\forall v \notin R : |R \cap N(v)| \neq 1$. (A similar notion is defined in Chapter 3 for the PDS problem.) Any solution needs to pick at least one node from any strong region R. To see this, observe that $\mathcal{P}^*(V \setminus R) = V \setminus R$; that is, even if we pick all nodes in $V \setminus R$ we cannot cover any other nodes. A straightforward application of the pigeonhole principle proves the following result.

Proposition 2.1.8 *The optimal value $\rho(G)$ is at least the maximum number of node-disjoint strong regions of G.*

Now we show an application of the above proposition. Let G be a $3 \times 3m$ grid. The three nodes in the first column of G covers all nodes, so we have $\rho(G) \leq 3$. We will show in Section 2.2 that this is an optimal solution (i.e., $\rho(G) = 3$). Consider the graph G' that is obtained from G by subdividing all the column edges (see Figure

2.3). The gray regions shown in the figure indicate a set of m node-disjoint strong regions; thus, by Proposition 2.1.8 we have $\rho(G') \geq m$.

Let G and H be two undirected graphs. It is shown in [29] that the Cartesian product of G and H has optimal value of at most $\min(\rho(G) \cdot |V(H)|, \rho(H) \cdot |V(G)|)$.

2.2 Path-width: A lower bound

In this section we prove that the path-width parameter is a lower bound for the optimal value of the Propagation problem on unweighted graphs. A straightforward application provides us with a tight bound for $\rho(G)$ on grids, and also shows that $\rho(G)$ is linear in $|V(G)|$ for *node-expander* graphs. We use standard terms and definitions pertaining to tree-width and path-width, see Appendix A or Diestel's book (Chapter 12 in [23]).

Theorem 2.2.1 *Let G be a graph. Then we have $\mathit{pw}(G) \leq \rho(G)$.*

Proof: Let S be a solution to the Propagation problem on G; that is, $\mathcal{P}^*(S) = V(G)$. By applying rule (R1) the set S is inserted into $\mathcal{P}(S)$. Assume that the remaining nodes are inserted in $\mathcal{P}(S)$ in the order $v_1, v_2, \cdots, v_m$ by applying the propagation rule sequentially on the nodes $u_1, u_2, \cdots, u_m$; that is, the propagation rule is applied in the sequence $u_1 \to v_1, u_2 \to v_2, \cdots, u_i \to v_i, \cdots, u_m \to v_m$. Based on this sequence, we construct a path decomposition $X_1, X_2, \ldots$ of G of width $|S|$ by defining the bags (or node sets) as follows: $X_1 = S \cup \{v_1\}$, $X_2 = (X_1 \setminus \{u_1\}) \cup \{v_2\}, \cdots, X_{i+1} = (X_i \setminus \{u_i\}) \cup \{v_{i+1}\}, \cdots$. We orient the edges in the same way that the propagation rule applies. If v is covered by applying the propagation rule to u (i.e., $u \to v$), then we orient the edge $\{u, v\}$ from u to v. Otherwise, we leave the edge unoriented. Note that whenever we remove a node u_i, say $u = u_i$, from a bag X_i to get the next bag X_{i+1}, then (in the setting of the propagation problem) u and all its neighbors have been already covered; that is, the closed neighborhood of u is contained in $\mathcal{P}(S)$. We claim that the sequence $X_1, X_2, \cdots$ is a path decomposition for G.

First we show that condition (T1) of the path decomposition is satisfied. We need to show that both end nodes of each edge are contained in some bag X_i. Consider an edge $e = \{u, v\} \in E(G)$. The edge e is either oriented $((u, v))$ or unoriented $(\{u, v\})$. There are three cases to consider.

1. If e is oriented then this edge is contained in the bag X_i corresponding to the step where v is covered (added to X_i); note that u and v are both in X_i.

2. If u and v are both sources of the valid orientation, then both are in X_1.

3. Otherwise, assume that e is unoriented and u is covered at step i and v is covered at a later step j, $j > i$. This implies that v is added to X_j at step j and

u is added to X_i at step i. Also note that u is not removed before step $(j+1)$, since u can be removed after it propagates and covers a neighbor and this propagation cannot happen before v is covered (by rule (R2)); as mentioned above we only remove u when all neighbors of u are covered. Therefore, u and v are both present in the same set X_j.

Hence, the given decomposition $X_1, X_2, \cdots, X_m$ satisfies property (T1) of the path decomposition.

Let $1 \leq i < j < k \leq m$. It can be checked that $X_i = (S \cup \{v_1, \cdots, v_i\}) \setminus \{u_1, \cdots, u_{i-1}\}$ and $X_i \cap X_k = (S \cup \{v_1, \cdots, v_i\}) \setminus \{u_1, \cdots, u_{k-1}\}$. This implies that $X_i \cap X_k \subseteq X_j$ and shows that property (T2) also holds. Hence, $\mathcal{X} = (X_1, \cdots, X_m)$ is a path decomposition with width equal to $|X_1| - 1 = |S \cup \{v\}| - 1 = |S|$, so the path-width of G is at most $\rho(G)$. $\square$

Unfortunately, the lower bound is weak, in general. Consider a caterpillar graph L_m shown in Figure 2.2. The path-width of L_m is $O(1)$. We claim that $\rho(L_m) = m$. Note that the two leaf nodes in each column form a strong region, so there are m node-disjoint strong regions; the strong regions are indicated by the gray regions in Figure 2.2. Hence, by Proposition 2.1.8 we have $\rho(L_m) \geq m$. A solution of size m is indicated by the boxed nodes in the figure; thus, $\rho(G) = m$. This shows that the path-width parameter may be a factor of $\Theta(|V|)$ smaller than $\rho(G)$. Figure 2.3 gives another graph to show this fact. The graph G is obtained from a $3 \times 3m$ grid by subdividing all column edges. In Section 2.1 after Proposition 2.1.8 we showed that $\rho(G) \geq m$. We claim that the path-width of G is $O(1)$, in fact, we have $\texttt{pw}(G) \leq 9$. Let $\mathcal{X} = (X_1, X_2, \ldots)$ be a sequence of node subsets, where X_i contains all nodes from the ith and the $(i+1)$th columns of G; thus, $|X_i| = 10$. The sequence $\mathcal{X}$ is a path decomposition of width $|X_i| - 1 = 9$; thus, $\texttt{pw}(G) \leq 9$.

Nevertheless, Theorem 2.2.1 allows us to get good bounds on $\rho(G)$ in some classes of graphs.

Corollary 2.2.2 *Let G be an $n_1 \times n_2$ grid. Then we have $\rho(G) = \min\{n_1, n_2\}$.*

Proof: Assume that $n_1 \leq n_2$. It is easy to check that all nodes in the first column propagate and cover all nodes in G, so $\rho(G) \leq n_1$. It is known that the tree-width of an $n_1 \times n_2$ grid is equal to $\min\{n_1, n_2\}$ [66], and the path-width of any graph is $\geq$ its tree-width (this follows directly from the definitions, since every path decomposition is a special tree decomposition). Therefore, Theorem 2.2.1 shows that $\rho(G) \geq n_1$. $\square$

Using the path-width parameter we can show that for expander graphs, $\rho(G)$ is linear in the number of nodes. The node-expansion of a graph G, denoted by α_G, is defined as

$$\alpha_G = \min_{1 \leq |S| \leq n/2} \frac{|N(S) \setminus S|}{|S|}$$

Proposition 2.2.3 *Let $G = (V, E)$ be a graph with node-expansion $\alpha = \alpha_G$ then we have:*

$$\rho(G) \geq \mathtt{pw}(G) \geq \frac{\alpha n - 2}{2(\alpha + 1)}.$$

Proof: Consider any path decomposition $\mathcal{X} = (X_1, X_2, \cdots, X_\ell)$ of width $\mathtt{pw}(G)$. There is a bag X_i such that $U = (\bigcup_{j=1}^{i-1} X_j) \setminus X_i$ has size at least $\frac{n}{2} - \mathtt{pw}(G)$. Removing the bag X_i separates the nodes in the first $i-1$ bags from the remaining nodes in G. Therefore, all neighbors of U (not in set U) are in X_i, so the node expansion of U is at most $\frac{\mathtt{pw}(G)+1}{n/2 - \mathtt{pw}(G)}$. Therefore, we get $\rho(G) \geq \mathtt{pw}(G) \geq \frac{\alpha n - 2}{2(\alpha+1)}$. □

2.3 Hardness of the propagation problem

In this section we provide our hardness results for the weighted PROPAGATION problem. First, in Subsection 2.3.1 we prove that the problem is NP-hard on planar graphs. Next, in Subsection 2.3.2 we prove that the PROPAGATION problem is NP-hard to approximate within ratio of $2^{\log^{1-\epsilon} n}$.

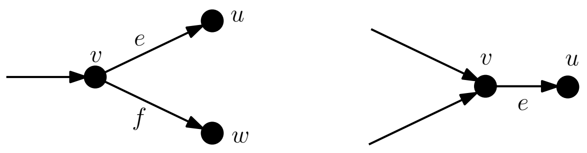
2.3.1 NP-hardness in planar weighted graphs

In this subsection we show that the PROPAGATION problem is NP-hard in planar graphs by a reduction from the DIRECTED HAMILTONIAN CYCLE problem in planar graphs. It has been proved [63] that the DIRECTED HAMILTONIAN CYCLE problem is NP-hard on planar digraphs of degree 3, that is the sum of the in-degree and out-degree for each node is 3.

Theorem 2.3.1 *The* PROPAGATION *problem is NP-hard in planar weighted graphs even when the weights of nodes are either zero or one.*

Theorem 2.3.2 ([63]) *The* DIRECTED HAMILTONIAN CYCLE *problem is NP-complete even in planar directed graphs where each node has either in-degree* 1 *and out-degree* 2 *or in-degree* 2 *and out-degree* 1.

Let G be an instance of the DIRECTED HAMILTONIAN CYCLE problem with the degree conditions as in Theorem 2.3.2. A node with in-degree 1 and out-degree 2 is called a type-α node, and a node with in-degree 2 and out-degree 1 is called a type-β node (see Figure 2.4). Let n_α denote the number of nodes of type-α in G. Starting from G, we will construct an instance of the PROPAGATION problem on a graph H such that G has a Hamiltonian cycle if and only if $\rho(H) = n_\alpha + 1$; in more detail, if G has no Hamiltonian cycle, then we will show that $\rho(H) > n_\alpha + 1$, otherwise, we will construct a solution S for the PROPAGATION problem on H with $\mathtt{W}(S) = n_\alpha + 1$ (see Lemma 2.3.4).



(a) Type-α node (b) Type-β node

Figure 2.4: Two different types of nodes

Gadgets: We have two different gadgets corresponding to the two types of nodes; see Figures 2.5 and 2.6(a). The nodes of each gadget are partitioned into *internal* nodes and *external* nodes. The external nodes of the gadgets are identified with the nodes of G. The internal nodes are local to the gadget, and they are only connected to the external nodes or to other internal nodes of the gadget. In our construction, we replace the out-going edges of each node v by the gadget corresponding to the type of v.

First, consider the type-β gadget. Let (v, u) be the unique out-going edge of a type-β node v in the graph G. The gadget has a path of length 3 from v to u, and also an edge from v to u (see Figure 2.5). The neighbor of v in the path, v_0, has a weight 0 and all other nodes have weight 1. The node v_0 has weight 0, so we may assume w.l.o.g. that v_0 is in our solution. Suppose that node v is covered; then we can apply the propagation rule to v_0 to cover v_1 and then to node v_1 to cover u. Now assume u is covered, but v is not covered yet. Then u and the internal node v_0 both have two non-covered neighbors v_1 and v. Hence, no propagation rule can be applied to either of them to cover v. This gadget models directed propagation in undirected graphs; u can be covered after v is covered. This gadget is also used in the construction of the type-α gadget.

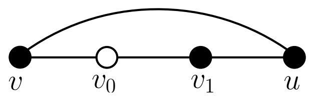


Figure 2.5: Gadget for type-β node; the nodes of weight zero (or, one) are indicated by white (or, black) circles.

Let v be a type-α node in G, and let $e = (v, u)$ and $f = (v, w)$ be the two out-going edges from v (see Figure 2.4(a)). The gadget for the type-α node is given in Figure 2.6(a). There are four directed edges $(x_e, y_e), (x_f, y_f), (z_e, u), (z_f, w)$ in this gadget; we replace each of them by a type-β gadget. The complete gadget for the type-α node is shown in Figure 2.6(b). All nodes in this gadget have weight 1 expect v_0 and the other four copies of v_0 (inside the four type-β gadgets corresponding to the four directed edges) which have weight 0. In the following discussion, we assume that all of the five nodes of weight 0 are picked and will not mention this explicitly. This gadget has some useful properties:

1. At least one of x_e, x_f or a copy of v_1 from the gadgets $(x_e, y_e), (x_f, y_f)$ should be picked. Suppose all nodes are picked except x_e, x_f and the copies of v_1 in gadgets (x_e, y_e) and (x_f, y_f). Then the propagation rule fails to apply to node v_0 since it has two non-covered nodes (namely, x_e, x_f). Similarly, it can be seen that the propagation rule fails to apply to the other nodes of the gadget to cover x_e, x_f. Note that picking x_e covers the same nodes as picking the copy of v_1 in the gadget (x_e, y_e), similarly x_f covers the same nodes as the copy of v_1 in the gadget (x_f, y_f). Hence, we may assume that we always pick a node from x_e or x_f, and never pick a copy of v_1.

2. Suppose that v is picked, and one of x_e or x_f, say x_e, is picked. Then by successive applications of the propagation rule we can cover u. (In detail: First y_e is covered through the β-gadget. Next, the propagation rule can be applied to y_e to cover z_e. Finally, u will be covered through the β-gadget.) On the other hand, if neither x_e nor x_f is picked and v is covered, then the propagation rule fails to apply to any node of this gadget to cover u.

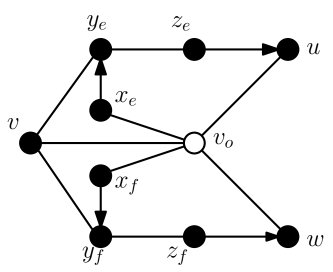
3. Suppose v is covered, and u, w are not covered. We claim that to cover both u and w through this gadget we need to pick at least 2 nodes of weight 1. From the previous property we know that at least one of x_e or x_f, say x_e, is picked. By picking x_e all nodes in the upper part of the type-α gadget are covered. The only propagation rule that may apply is to the node v to cover y_f, and this can occur only if all other neighbors of v outside of this gadget are covered. Now, we can check that all covered nodes have at least two non-covered neighbors; for example, y_f has 2 non-covered neighbors in the gadget (x_f, y_f) (see Figure 2.6(b)). Hence, no other propagation can occur. This shows that we need to pick at least one more node of weight 1 to be able to cover w.

4. Suppose u and w are covered, and only one of x_e, x_f is picked, say x_e. Then we claim that the only propagation that can occur is through the β-gadget (x_e, y_e), and this propagation covers y_e. It can be seen that no other propagation rule can be applied to any internal node to cover v.

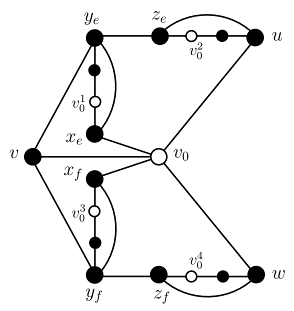
5. Suppose all three nodes v, u, w are covered, and at least one of x_e, x_f is picked, say x_e. Then all of the remaining nodes in this gadget will be covered. The propagation occurs in the following order. First, by applying the propagation rule to v_0 we cover x_f (the only non-covered neighbor of x_0). Next, the node y_f is covered through the β-gadget at (x_f, y_f). Next, by applying the propagation rule to y_f we cover z_f. Finally, w is covered through the β-gadget at (z_f, w).

The reduction: Now we construct a planar instance, H, of the PROPAGATION problem as follows.

1. Start from a copy of G.



(a)



(b) Complete gadget

Figure 2.6: Gadget for type-α node; the nodes of weight zero (or, one) are indicated by white (or, black) circles.

2. Let s be an arbitrary node of type-α, and let (t, s) be the incoming edge to s; note that this step applies to only one directed edge.

   (a) Make two new nodes s', t'. Remove the edge (t, s) and add $(t, t'), (s', s)$. Let G' be the obtained graph. (See Figure 2.7 for an illustration.) Note that G' has a Hamiltonian path from s' to t' if and only if G has a Hamiltonian cycle.

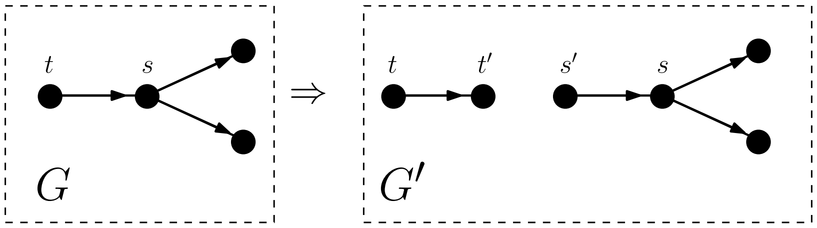


Figure 2.7: Modifying the edge (t, s) in G

   (b) Replace (s', s) by a type-β gadget.

3. For each type-β node v do the following:

   (a) Let $e = (v, u)$ be the unique out-going edge at v. Remove $e = (v, u)$ from G'.

   (b) Add a gadget as shown in Figure 2.5 between u and v. The neighbor of v labeled by v_0 has weight 0 and the other nodes have weight 1.

4. For each type-α node v (including the node s from step 2 above) do the following:

   (a) Remove the out-going edges $e = (v, u)$, $f = (v, w)$ from v.

(b) Add a gadget on seven new nodes x_e, y_e, z_e, v_0, x_f, y_f, z_f and connect them as shown in Figure 2.6(b) to the nodes v, u, w. The nodes in the gadget are disjoint from the remaining nodes in the graph.

5. Let H be the obtained graph.

The first property of the type-α gadgets proves the following result.

Lemma 2.3.3 *In each solution to the* PROPAGATION *problem on H, we need to have at least one internal node of weight* 1 *from each type-α gadget.*

Hence, every solution needs to have at least n_α nodes (of weight 1); by the first property of the type-α gadgets this node is either x_e or x_f. It can be checked that after picking these nodes no propagation can start, so $\rho(H) \geq n_\alpha + 1$. The following lemma completes the proof of Theorem 2.3.1.

Lemma 2.3.4 *The graph G has a Hamiltonian cycle if and only if $\rho(H) = n_\alpha + 1$.*

Proof: Note that the graph G' has a Hamiltonian path from s' to t' if and only if G has a Hamiltonian cycle, so it is enough to prove that G' has a Hamiltonian path from s' to t' if and only if $\rho(H) = n_\alpha + 1$.

Assume G' has a Hamiltonian path P from s' to t'. We define a solution S as follows. The set S contains the node s' and all nodes of weight 0 (from all gadgets). From each gadget corresponding to a type-α node v the set S contains the node x_e, where e is the out-going edge from v in the Hamiltonian path P. We show that by picking the nodes of S, all nodes of H can be covered.

Let $v_1 = s', v_2 = s, \ldots, v_j, \ldots, v_n = t'$ be the order of nodes in the Hamiltonian path P. First, we show that the nodes of G' are covered in H in the order that they appear in P; the proof is by an induction on the index of a node in P. The base case $j = 1$ is trivial, since $v_1 = s'$ is in S. Assume the claim is correct for $j \geq 1$, i.e., nodes $v_1, \ldots, v_j$ are covered. Consider the node v_{j+1} and assume that v_j is a node of type-α; note that $e = (v_j, v_{j+1})$ is a directed edge in P. By the second property of the type-α gadgets, the propagation in the gadget corresponding to the node v_j starts from the node x_e and finally covers v_{j+1}; note that $x_e \in S$. The case when v_j is a node of type-β is similar. This completes the induction step, and shows that all nodes of G' are covered in H. After all the original nodes from G' are covered, the remaining internal nodes in all gadgets will be covered; this follows from the last property of the type-α gadgets. This shows that S is a solution; hence, $\rho(H) \leq \mathtt{W}(S) \leq n_\alpha + 1$.

Assume H has a solution S of weight $n_\alpha + 1$. Note that S contains all nodes of weight 0 (i.e., all v_0's). Also S has one of the nodes x_e, x_f from each type-α gadget. We claim that the extra node of S should be from the type-β gadget at (s', s). Consider the gadget at (s', s), and note that there is no other gadget that has s' as an external node. By property of type-β gadget the node s' cannot be covered unless s' or v_1 is picked. Hence, we may assume that s' is picked; that is, $s' \in S$. Note that since S has weight $n_\alpha + 1$, we cannot have more than one internal node of weight 1 from each type-α gadget. Thus, s' is the only node from G' in the set S. Therefore, the node s' is the first node from G' that is covered in H. Next, the node s is covered through the type-β gadget installed between s' and s. In the following discussion we only focus on the original nodes from G' in the graph H. We consider the order in which nodes of G' are covered in H, and we claim that this order defines a Hamiltonian path P in G'. Let $v_1 = s', v_2 = s, v_3, \cdots$ be the order in which nodes of G' are covered. We prove by induction that the order $v_1, v_2, \cdots, v_j$ (for each $j \geq 1$) defines a directed path in G', and also v_{j+1} is covered after v_j (either through a type-α gadget or a type-β gadget). The base case $j = 1$ is trivial. Assume the claim is correct for $j \geq 1$. Consider the node v_{j+1}. We prove that v_{j+1} is covered through a gadget that we used to replace the out-going edges at v_j in the construction of H. Assume that v_{j+1} is covered through a gadget with external nodes v_i and v_{j+1}, for some i less than j. We claim that (v_i, v_{j+1}) is a directed edge in G; this follows from the properties of the type-β gadget and the 4th property of the type-α gadget. By the induction hypothesis, we know that (v_i, v_{i+1}) is a directed edge in G. Hence, v_i is a type-α node, and v_{i+1}, v_{j+1} are covered through the gadget with the external nodes v_i, v_{i+1}, v_{j+1}. This is not possible, since, by the 3rd property of type-α gadgets, it requires to pick at least two nodes of weight 1. Therefore, v_{j+1} can only be covered through the gadget containing the external nodes v_j, v_{j+1}. This gadget is either a type-β gadget or a type-α gadget; both cases imply that (v_j, v_{j+1}) is a directed edge in G. This completes the induction step and shows that $v_1 = s', v_2, \ldots, v_j, v_{j+1}, \ldots, v_n = t'$ is a directed Hamiltonian path in G'. □

2.3.2 Hardness of approximation in weighted graphs

In this subsection we prove that it is NP-hard to approximate the weighted PROPAGATION problem within a factor of $2^{\log^{1-\epsilon} n}$. For expository reasons, we first prove Theorem 2.3.5 that gives a hardness result with a weaker complexity assumption of NP $\not\subseteq$ DTIME$(n^{polylog(n)})$. Our main result in this section, Theorem 2.3.8, does not depend on Theorem 2.3.5, but the proof of the main result is an easy extension of the proof of the weaker result. Experts in the area may prefer to skip Theorem 2.3.5 and its proof.

Theorem 2.3.5 *The weighted* PROPAGATION *problem cannot be approximated within ratio* $2^{\log^{1-\epsilon} n}$, *for any fixed* $\epsilon > 0$, *unless* NP $\subseteq$ DTIME$(n^{polylog(n)})$.

In the MINREP [44] problem we are given a bipartite graph $G = (A, B, E)$ with a partition of A and B into subsets. Let q_A and q_B denote the number of sets in the partition of A and B, respectively. Let $A = A_1 \cup A_2 \cup \cdots \cup A_{q_A}$ denote the partition of A, and let $B = B_1 \cup B_2 \cup \cdots \cup B_{q_B}$ denote the partition of B. This partition naturally defines a super bipartite graph $\mathcal{H} = (\mathcal{A}, \mathcal{B}, \mathcal{E})$. The super nodes of $\mathcal{H}$ are $A_1, A_2, \ldots, A_{q_A}$ and $B_1, B_2, \ldots, B_{q_B}$. There is a super edge between super nodes A_i and B_j if there exists some $a \in A_i$ and $b \in B_j$ such that ab is an edge in G. We say that a super edge A_iB_j is covered by nodes a, b if $a \in A_i$, $b \in B_j$, and there is an edge between a and b in G. Given $S \subseteq A \cup B$ we say that the super edge A_iB_j is covered by S if there exists $a, b \in S$ that covers A_iB_j. The goal in the MinRep problem is to pick a minimum-size set of nodes, $A' \cup B' \subseteq V(G)$, to cover all super edges in $\mathcal{H}$. Note that we need a pair of nodes to cover a super edge, and the pair should induce an edge between the two super nodes of the super edge; moreover, a node in $A' \cup B'$ may be useful for covering more than one super edge. The following Theorem is from [44].

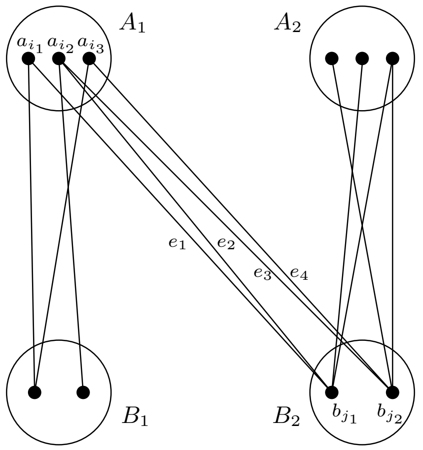


Figure 2.8: MinRep instance G

Theorem 2.3.6 (Theorem 5.4 in [44]) *The* MinRep *problem cannot be approximated within ratio* $2^{\log^{1-\epsilon} n}$*, for any fixed* $\epsilon > 0$*, unless* $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$.

The reduction:

1. Start from a complete bipartite graph with parts Z and Y, where $|Z| = |A \cup B| + 1$ and $|Y| = 2|\mathcal{E}|$. Corresponding to each node $a \in A \cup B$ we have a node $\hat{a}$ of weight 1 in Z-part, and corresponding to each super edge $A_iB_j \in \mathcal{E}$ we have two nodes in the Y-part labeled by $d'(A_i, B_j)$ and $d''(A_i, B_j)$ each of weight 2. Also there is an extra node v_0 of weight 0 in the Z-part.

2. In the next step we "install" several copies of the gadget shown in Figure 2.9(a). The nodes labeled by x and y are called the *internal* nodes, and the rest of nodes are called the *external* nodes. In all the copies of the gadget, the node x has weight 0, the nodes $\hat{a}$ and $\hat{b}$ have weight 1, and the rest of nodes have weight 2.

3. For each edge $\{a, b\}$ of $E(G)$ do the following:

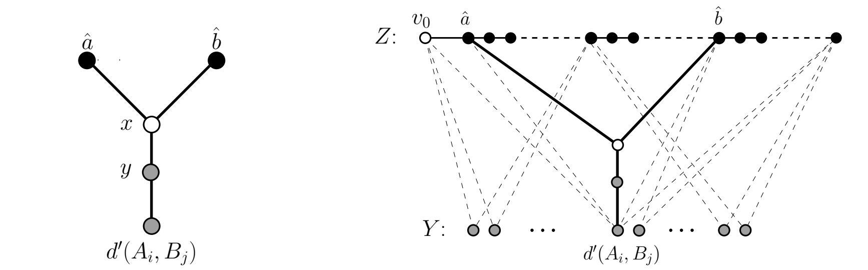


Figure 2.9: Hardness construction; the nodes of weight 0, 1, and 2 are indicated by white, black, and gray circles, respectively.

(a) Suppose that $a \in A_i$ and $b \in B_j$; note that A_iB_j is a super edge in $\mathcal{H}$.

(b) Make a new copy of the gadget in Figure 2.9(a) on nodes $\hat{a}$, $\hat{b}$, and $d'(A_i, B_j)$, and make another copy of the gadget on nodes $\hat{a}$, $\hat{b}$, and $d''(A_i, B_j)$.

4. Add edges between the nodes in the Z-part such that the graph induced on the Z-part is a path starting from the node v_0; the order of other nodes in this path is arbitrary and not important.

5. Let H be the obtained graph (see Figure 2.9(b) for an illustration).

Analysis: We may assume that all nodes of weight 0 are picked in any solution. We may not mention this explicitly in the rest of the proof. Consider the gadget shown in Figure 2.9(a) and discussed in step (2) above. By picking the nodes $\hat{a}, \hat{b}$ all nodes in this gadget can be covered; in detail, the propagation rule is applied in the sequence $x \rightarrow y, y \rightarrow d'(A_i, B_j)$. We may assume no node of weight 2 is picked from this gadget in any optimal solution, since we can replace it by the nodes $\hat{a}$, $\hat{b}$ and get a solution of no heavier weight. This is an important property, and it implies that there is an optimal solution with nodes from the Z-part. The following lemma completes the proof of Theorem 2.3.5.

Lemma 2.3.7 *Assume that the* MinRep *instance $\mathcal{H}$ has an optimal solution of size W^*. Then $W^* - 1 \leq \rho(H) \leq W^*$.*

Proof: First, we prove that $\rho(H) \leq W^*$. Assume that $A^* \cup B^*$ is a solution for the MinRep instance G. We claim that by picking the nodes of $\{\hat{a} | a \in A^* \cup B^*\}$ and all nodes of weight 0 we get a solution to H. To see this consider a super edge A_iB_j of $\mathcal{H}$. The set $A^* \cup B^*$ covers all super edges in $\mathcal{H}$. Hence, there exists a pair of nodes $a \in A^* \cap A_i$, $b \in B^* \cap B_j$ that induces an edge in G. Consider the cover testing gadget corresponding to the edge $\{a, b\}$ which has external nodes $\hat{a}$, $\hat{b}$, and $d'(A_i, B_j)$; this is the first copy of the cover testing gadget corresponding to the edge $\{a, b\}$. All nodes in this gadget are covered; in detail, the propagation rule is applied in the sequence $x \to y$, $y \to d'(A_i, B_j)$. Similarly the node $d''(A_i, B_j)$ is covered through the second gadget corresponding to the edge $\{a, b\}$. Hence, both nodes corresponding to the super edge A_iB_j are covered. This is true for all super edges, so all nodes in Y are covered. Now, starting from v_0 we can cover all nodes in Z. Note that the graph induced on the Z-part is a path and all neighbors of nodes in Z are either picked (weight 0 nodes) or covered (nodes in Y). Finally all of the remaining non-covered nodes (of weight 2) inside the gadgets are covered; note that at this point all nodes in the Z-part are covered. This shows that we have a solution of weight $|A^* \cup B^*|$ to the instance H of the PROPAGATION problem. Therefore, we have $\rho(H) \leq W^*$.

Now, we prove that $W^* - 1 \leq \rho(H)$. Let S^* be an optimal solution to the PROPAGATION problem on H of weight $\rho(H)$, and assume for the sake of the contradiction that $\rho(H) \leq W^* - 2$. We may assume that S^* has no node of weight greater than 1. To see this note that if a node of weight 2 from any copy of the cover testing gadget is picked, then we may replace it by the two external nodes of the gadget in the Z-part. Note that taking these two external nodes covers all nodes in this gadget independently of the other picked nodes in S^*. This replacement gives a solution of no heavier weight than S^*. We may also assume that all nodes of weight 0, including v_0, are in S^*. A cover testing gadget corresponding to the edge $\{a, b\}$ is called a *good* gadget if both $\hat{a}, \hat{b}$ are in S^*, otherwise it is called a *bad* gadget. We claim that by taking S^* only the nodes of the good gadgets are covered, and no propagation can occur in any bad gadgets. This is a contradiction, since S^* is a solution. Therefore, proving this claim shows that $W^* - 1 \leq \rho(H)$.

Let $\overline{S}$ denotes the set of nodes of weight 1 in S^*; that is, $\overline{S} = \{a \in A \cup B | \hat{a} \in S^*\}$. Note that $\left|\overline{S}\right| = \mathtt{W}(S^*) \leq W^* - 2$, so $\overline{S}$ is not a solution to the MINREP instance $\mathcal{H}$. By taking S^*, it is obvious that all nodes of the good gadgets are covered, so we only need to show that no more propagation can be applied. Note that internal nodes in the good gadgets have no non-covered neighbors, and the nodes labeled by y in the bad gadgets are not covered. Hence, the propagation rule may apply only to the following three type of nodes: 1) nodes in the Z-part, 2) nodes in the Y-part, 3) internal nodes of weight 0 in the bad gadgets.

- Let v be a node in the Z-part. Note that $\overline{S}$ is not a solution to the MINREP instance, so there is at least one super edge A_iB_j that is not covered by $\overline{S}$. Hence, the two nodes $d'(A_i, B_j), d''(A_i, B_j)$ corresponding to this super edge are not covered yet. Therefore, the propagation rule cannot be applied to v, since it has two non-covered neighbors in the Y-part; recall that in H there is a complete bipartite graph between the Y-part and the Z-part.

- Let v be a node in the Y-part. Note that $\left|\overline{S}\right| = \rho(H) \leq W^* - 2 \leq |A \cup B| - 2$, so there are at least two nodes of weight 1 in the Z-part that are not in S^*. Hence, v has at least two non-covered neighbors, and the propagation rule cannot be applied to it.

- Let x be a node of weight 0 in a bad gadget with external nodes $\hat{a}, \hat{b}$. Assume that $\hat{b}$ is the non-picked node. The node x has at least two non-covered neighbors $y, \hat{b}$, so the propagation rule cannot be applied to x.

Hence, after all nodes in the good gadgets are covered, no other propagation can occur to cover any one of the remaining non-covered nodes. This proves the above claim. Therefore, we have $W^* - 1 \leq \rho(H)$. □

Improved hardness result

In this subsection, we give a reduction from the Extended Label Cover problem to the Propagation problem that improves on the hardness result of Theorem 2.3.5. The improvement is obtained by starting from a stronger result (compare Theorem 2.3.9 with Theorem 2.3.6) and by using an extension of the gadget used in the previous proof.

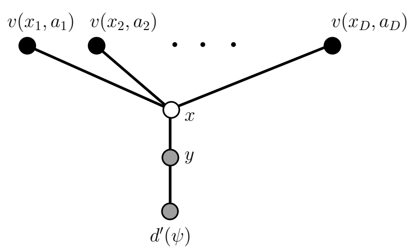
Theorem 2.3.8 *For any fixed $\epsilon > 0$, it is NP-hard to approximate the weighted* Propagation *problem within ratio $2^{\log^{1-\epsilon} n}$.*

In the *constraint satisfaction* problem we are given $\mathcal{C} = (\Psi, X, F, D, R)$ where X is a set of n variables defined on a field F, and Ψ is a set of m constraints defined on X. Each constraint $\psi \in \Psi$ depends on D variables, and for each ψ we are also given a set $R_\psi \subseteq F^D$ of the satisfying assignments. For each variable $x \in X$, we are also given an integer weight $\mathtt{W}(x)$. Let $\overline{W} = \mathtt{W}(X)$. A solution to $\mathcal{C}$ is a function $f : X \to 2^F$ that assigns a set $f(x) \subseteq F$ to each variable $x \in X$. The weight of a solution f is defined by $\sum_{x \in X} \mathtt{W}(x) \, |f(x)|$. We say that a constraint ψ defined on variables $x_1, \ldots, x_D$ is satisfied in f if there exists a satisfying assignment $(a_1, \ldots, a_D) \in R_\psi$ such that $a_i \in f(x_i)$, for all $i = 1, \ldots, D$. The goal in the Extended Label Cover [19] problem is to find a minimum-weight solution that satisfies all the constraints. For example, consider the version of the SAT problem with exactly 3 literals per clause. Then $D = 3$, and the field is given by $F = \{0, 1\}$. For each clause ψ, R_ψ consists of the 7 truth assignments to the 3 variables of ψ that satisfy ψ; for example, for $\psi = x_1 \vee x_2 \vee x_3$, R_ψ consists of the 7 triples in $F^3 \setminus \{(0,0,0)\}$. In the following, we assume that, in the given instance $\mathcal{C}$ of the Extended Label Cover problem, $\overline{W}$ and $|F|^D$ are polynomial in m.

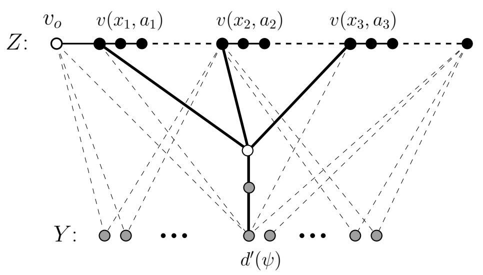
Theorem 2.3.9 ([19]) *For any fixed $\epsilon > 0$, it is NP-hard to approximate the* Extended Label Cover *problem within the ratio of $2^{\log^{1-\epsilon} m}$, even when $\overline{W}, |F|^D, n \leq$ poly(m).*

The reduction:

1. Start from a complete bipartite graph with parts Z and Y, where $|Z| = n\,|F| + 1$ and $|Y| = 2m$. Corresponding to each constraint $\psi \in \Psi$ we have



(a) cover testing gadget



(b) Structure of the graph H

Figure 2.10: Hardness construction

two nodes in the Y-part labeled by $d'(\psi)$ and $d''(\psi)$ of weight $\overline{W} = \mathtt{W}(X)$. We have a node $v(x, a)$ of weight $\mathtt{W}(x)$ in the Z-part corresponding to each variable $x \in X$ and each $a \in F$; the node $v(x, a)$ represents the assignment of a to the variable x in the constraint satisfaction problem. Also there is an extra node v_0 in the Z-part of weight 0.

2. In the next step we "install" several copies of the gadget shown in Figure 2.10(a). The nodes labeled by x and y are called the *internal* nodes, and the rest of the nodes are called the *external* nodes. In all the copies of the gadget, the node x has weight 0, the node $v(x_i, a_i)$ (for $i = 1, \ldots, D$) has weight $\mathtt{W}(x_i)$, and the rest of the nodes have weight $\overline{W}$.

3. For each constraint $\psi \in \Psi$ of $\mathcal{C}$ and for each satisfying assignment $(a_1, \ldots, a_D)$ of ψ do the following:

   Make a new copy of the gadget in Figure 2.9(a) on nodes $v(x_i, a_i)$ (for $i = 1, \ldots, D$) and $d'(\psi)$, and make another copy of this gadget on nodes $v(x_i, a_i)$ (for $i = 1, \ldots, D$) and $d''(\psi)$.

4. Add edges between the nodes in the Z-part such that the graph induced on the Z-part is a path starting from the node v_0.

5. Let H be the obtained graph (see Figure 2.9(b) for an illustration).

Analysis: Note that the size of H is polynomial in m. To prove the claimed hardness threshold for the Propagation problem, we need to prove the following lemma. Let $\mathtt{W}_{max} = \max_{x \in X} \mathtt{W}(x)$.

Lemma 2.3.10 *Assume that the extended label cover instance $\mathcal{C}$ has an optimal solution of weight W^*. Then $W^* - \mathtt{W}_{max} \leq \rho(H) \leq W^*$.*

Proof: The proof is similar to Lemma 2.3.7. Here we only provide a sketch. Let $f : X \to 2^F$ be an optimal solution to the instance $\mathcal{C}$ of weight $W^* = \sum \mathtt{W}(x) \, |f(x)|$.

We claim that by picking the set of nodes $\{v(x,a)|x \in X, a \in f(x)\}$, and all nodes of weight zero we get a solution to H. This proves that $\rho(H) \leq W^*$.

Let S^* be an optimal solution to the Propagation problem on H, and assume for the sake of contradiction that $\rho(H) = \mathtt{W}(S^*) \leq W^* - \mathtt{W}_{max} - 1$. We may assume that all nodes of weight 0 are picked, and no node of weight $\overline{W}$ is in S^*; this is proved by replacing a node of weight $\overline{W}$ with nodes from the Z-part as in the proof of Lemma 2.3.7. A gadget is called a good gadget if all of its nodes in the Z-part are picked; that is, they are in S^*. We claim that S^* can only cover nodes of the good gadgets. The proof is similar to the same claim in Lemma 2.3.7; this is proved by considering three different types of nodes in H and showing that no propagation rule applies. Define a function $\overline{f}$ by $\overline{f}(x) = \{a \in F|v(x,a) \in S^*\}$, $\forall x \in X$. Note that $\overline{f}$ is not a solution to $\mathcal{C}$, since the weight of $\overline{f}$ is less than W^*. Hence, there are some bad gadgets in H. Therefore, some nodes in H cannot be covered by picking S^*. This is a contradiction, since S^* is a solution to H. Thus, we have $W^* - \mathtt{W}_{max} \leq \rho(H)$. □

Bellare et al. [9] made the following conjecture pertaining to PCPs': NP has PCPs with a constant number of queries, logarithmic number of random bits, proof table entries defined over a field F of polynomial size, perfect completeness and polynomially small soundness. Chuzhoy and Khanna [19] noticed that if this conjecture is correct, then it is NP-hard to approximate the Extended Label Cover problem within the ratio of $m^{\Omega(1)}$, where $m = |\Psi|$. Thus, modulo this conjecture, the same reduction and analysis as above imply that it is NP-hard to approximate the weighted Propagation problem within ratio of $n^{\Omega(1)}$.

2.4 Further discussion

2.4.1 Parallel propagation and the diameter

In this subsection, we first introduce the *parallel* propagation rule. In some applications it is important to guarantee that the propagation can be done in at most ℓ "parallel" rounds by picking k initial nodes. Informally speaking, in each "parallel" round we apply the propagation rule to all covered nodes where the propagation rule can be applied. Given the parameters k and ℓ, we show that the maximum number of nodes of such a graph is at most $k \times \ell$. Finally, we show that the diameter of a graph and the number of parallel rounds needed to cover the graph are not closely related.

Given a graph $G = (V, E)$ and a subset of nodes $S \subseteq V$, the set of nodes that can be covered by applying ℓ rounds of parallel propagation, denoted by $\mathcal{P}^\ell(S)$, is defined recursively as follows:

$$\mathcal{P}^\ell(S) = \begin{cases} S & \ell = 0 \\ \mathcal{P}^{\ell-1}(S) \bigcup \{v : \{u,v\} \in E, N[u] \setminus \{v\} \subseteq \mathcal{P}^{\ell-1}(S)\} & \ell \geq 1 \end{cases}$$

Given a solution S to the PROPAGATION problem, we say that S covers V in ℓ parallel rounds of propagation if $\mathcal{P}^{\ell}(S) = V(G)$. Given a parameter ℓ, we can ask to find a minimum-size set of nodes that covers all nodes in ℓ parallel rounds of propagation; this problem is called the ℓ-round PROPAGATION problem, and it is studied in Chapter 4.

Proposition 2.4.1 *Assume that $G = (V, E)$ has a solution of size k that can cover V in ℓ parallel rounds of propagation. Then we have $|V| \leq k\ell$. Moreover, a $k \times \ell$ grid achieves this maximum.*

Proof: Consider a solution S of size k. Each node from S initiate a "chain" of propagation of length at most ℓ; the order in which nodes are covered (in each chain) defines a path of length ℓ in G. Therefore, we have $|V(G)| \leq k \cdot \ell$.

Let G be a $k \times \ell$ grid. The k nodes in the first column can cover all nodes of G in ℓ parallel rounds of propagation. In the first round, the first column is covered. In the second round, by applying the propagation rule to all nodes in the first column all nodes in the second column will be covered. This can continue, and all nodes of G can be covered in ℓ parallel rounds. This shows that a $k \times \ell$ grid achieves the maximum number of nodes given in the proposition. $\square$

Proposition 2.4.2 *There exist a planar graph on n nodes with diameter $\Theta(\sqrt{n})$ that has an optimal solution (for the* PROPAGATION *problem) that needs $\Theta(n)$ parallel rounds to be propagated.*

Proof: Start from a $\sqrt{n} \times \sqrt{n}$ grid and modify it as shown in Figure 2.11 to get a planar graph G on n nodes (the figure shows the modified 7×7 grid). The graph G has diameter $\Theta(\sqrt{n})$. It is easy to see that G has the $\sqrt{n} \times \sqrt{n}$ grid as a subgraph (so as a minor). Therefore, the path-width of G is at least $\sqrt{n}$, and we have: $\rho(G) \geq \sqrt{n}$. It is easy to check that the first column can cover G. In the first round all nodes in the first column are covered. The only node in the first column that has a unique non-covered neighbor is the node in the first row and all other nodes in the first column have 2 non-covered neighbors. Hence, by applying the propagation rule, we can cover the first node in the second column. Now the second node in the first column has a unique non-covered neighbor, so we can apply the propagation rule to it. We can check that we can continue and cover the nodes in the second column, covering one node per round. Therefore, the nodes in the second column can be covered in $\sqrt{n}$ rounds. The same steps apply for the other columns. Hence, we need $\sqrt{n}$ parallel rounds to cover each column. Hence, we need $\Theta(n)$ parallel rounds to cover the whole graph G. $\square$

In the above Proposition, we show that for one particular optimal solution of the PROPAGATION problem we need $\Theta(n)$ number of parallel rounds to cover the whole graph. It seems that the same statement is valid for all optimal solutions. Computational experiments on (the modified) $k \times k$ grids for $k \leq 6$ shows that any optimal solution contains only nodes from either the first two columns or the

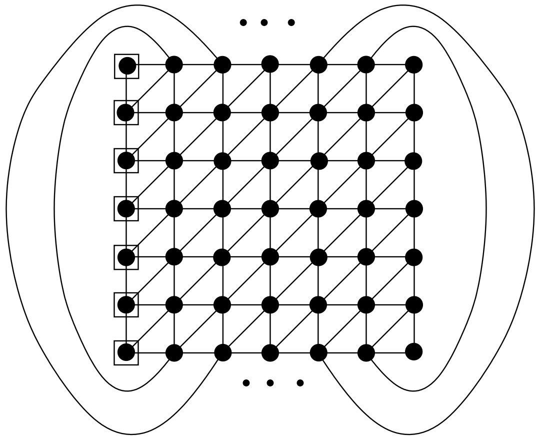


Figure 2.11: The modified 7×7 grid

last two columns; that is, there is no optimal solution picking a node from columns $3, \ldots, k-3$. Moreover, all optimal solutions need exactly $k(k-1)$ parallel rounds to cover the whole graph; note that this is equal to the number of nodes not picked in any optimal solution. On the other hand, if we pick more nodes than optimal solution, then it is possible to reduce the number of parallel rounds needed to cover the whole graph; for example, by picking the nodes in the first row and the last row of the $k \times k$ grid we can cover the whole graph in $\Theta(k)$ parallel rounds.

Now, we show another graph with $\Theta(n)$ diameter which has an optimal solution that can be propagated in $O(1)$ parallel rounds. Consider the Caterpillar graph L_m shown in Figure 2.2. This graph has $3m$ nodes, and it has diameter $m+1$. Note that the gray regions shown in Figure 2.2 indicate a set of m node-disjoint strong regions, so by Proposition 2.1.8 we have $\rho(L_m) \geq m$. The boxed nodes in the figure shows an optimal solution of size m that can be propagated in 2 parallel rounds to cover the whole graph.

Proposition 2.4.3 *There exist a planar graph on n nodes with diameter $\Theta(n)$ that has an optimal solution that can be propagated in $O(1)$ parallel rounds.*

2.4.2 A conjecture on graph product

Let P_m denote a path on m nodes, and let $G = P_{n_1} \times P_{n_2} \times \cdots \times P_{n_d}$ (where $n_1 \leq n_2 \leq \cdots \leq n_d$). The graph G is obtained from n_d disjoint copies of $P_{n_1} \times P_{n_2} \times \cdots \times P_{n_{d-1}}$ by connecting nodes in the ith copy to the corresponding nodes in the $(i+1)$th copy for each $i \in \{1, 2, \ldots, d-1\}$; there is a copy corresponding to each node of P_{n_d}. One can check that there is a solution of size $n_1 \times n_2 \times \cdots \times n_{d-1}$ to the Propagation problem on G, by picking all nodes in the copy of $P_{n_1} \times P_{n_2} \times \cdots \times P_{n_{d-1}}$ in G corresponding to an end node of the path P_{n_d}. We conjecture that we cannot do better than this; note that Corollary 2.2.2 proves this for the special case of $d = 2$. We have confirmed this conjecture on small instances $((n_1, n_2, n_3) = (2, 3, 3), (2, 3, 4), (3, 3, 3), (3, 3, 4))$ via computational experiments. The authors in [29] have shown that the size of the optimal solution for the d-dimensional hypercube is 2^{d-1}. This result is a special case of our conjecture on the Cartesian product of paths.

Conjecture 2.4.4 *Suppose that G is the Cartesian product of paths of lengths $n_1, n_2, \ldots, n_d$, where $n_1 \leq n_2 \leq \cdots \leq n_d$, then we have $\rho(G) = n_1 \times n_2 \times \cdots \times n_{d-1}$.*

Chapter 3

The power dominating set problem

Our focus in this chapter is on the Power Dominating Set (abbreviated as PDS) problem. Power domination is defined by two rules; the first rule is the same covering rule as in the Dominating Set problem, and the second rule is the same propagation rule as in the Propagation problem. More precisely, given a graph $G = (V, E)$ and a set of nodes S, the set of nodes that are *power dominated* by S, denoted $\mathcal{P}^*(S)$, is obtained as follows.

(R1) if node u is in S, then u and all of its neighbors are added to $\mathcal{P}(S)$;

(R2) (propagation) if node u is in $\mathcal{P}(S)$, one of its neighbors v is not in $\mathcal{P}(S)$, and all other neighbors of u are in $\mathcal{P}(S)$, then v is added to $\mathcal{P}(S)$.

The set $\mathcal{P}(S)$ changes as we apply (R1) and (R2); the final set $\mathcal{P}(S)$ is denoted by $\mathcal{P}^*(S)$. The PDS problem is to find a node-set S of minimum size that power dominates all nodes (i.e., find $S \subseteq V$ with $|S|$ minimum such that $\mathcal{P}^*(S) = V$). We denote the size of an optimal solution for the PDS problem on the graph G by $\gamma(G)$. Given a weight function, $\mathtt{W} : V \to \mathbb{Q}$, defined on nodes of a graph G, we can also ask to find a set S with minimum weight that covers all nodes of G. We denote the weight of an optimal solution by $\gamma(G, \mathtt{W})$. When the weight function $\mathtt{W}$ is clear from the context, we simply denote the weight of an optimal solution by $\gamma(G)$. Given a set of nodes S, we say that a node v is *picked* if v is in the set S. When a node v is added to $\mathcal{P}(S)$ by applying rule (R1) (rule (R2)) to node u ($\neq v$), we say that v is *power dominated* by applying rule (R1) (rule (R2)) to u and we denote this by $u \to v$. Rule (R2) is also called the propagation rule.

Consider the graph in Figure 2.1. The minimum power dominating set has size one – if S has any one node of the innermost triangle (like v), then $\mathcal{P}^*(S) = V$. In more detail, we apply rule (R1) to v to power dominate all the nodes of the innermost triangle and one node of the second triangle; then by two applications of rule (R2) (to each of the nodes in the first triangle not in S), we power dominate the other two nodes of the second triangle; then by three applications of (R2) (to each of the nodes in the second triangle) we power dominate all three nodes of the third triangle; etc.

The PDS problem arose in the context of electric power networks, where the aim is to monitor all of the network by placing a minimum-size set of very expensive devices called phase measurement units; these units have the capability of monitoring remote elements via propagation (as in rule (R2)); see Brueni [13], Baldwin et al. [8], and Mili et al. [54]. In the engineering literature, the problem is called the PMU placement problem.

Our motivation comes from the area of approximation algorithms and hardness results. The Dominating Set problem is a so-called *covering problem*; we wish to cover all nodes of the graph by choosing as few node neighborhoods as possible. In fact, the Dominating Set problem is a special case of the well-known Set Covering problem. In the latter problem, we are given a family of sets on a groundset, and the goal is to find the minimum number of sets whose union equals the groundset.

Such covering problems have been extensively investigated. One of the key positive results dates from the 1970's, when Johnson [37], Lovász [49] and later Chvátal [20] showed that the greedy method achieves an approximation guarantee of $O(\log |V|)$ where $|V|$ denotes the size of the ground set, see also [70]. These algorithms provide the same approximation guarantee for the Dominating Set problem, as the Dominating Set problem is a special case of the Set Covering problem. Several negative results (on the hardness of approximation) have been discovered over the last few years: Lund and Yannakakis [50] showed that the Set Covering problem is hard to approximate within a ratio of $\Omega(\log n)$ and later, Feige [25] showed that it is hard to approximate within a ratio of $(1-\epsilon)\ln n$, modulo some variants of the $\mathsf{P} \neq \mathsf{NP}$ assumption. The Dominating Set problem has the same hardness of approximation as the Set Covering problem (see [25]).

A natural question is what happens to covering problems (in the setting of approximation algorithms and hardness results) when we augment the covering rule with a propagation rule. PDS seems to be a key problem of this type, since it is obtained from the Dominating Set problem by adding a simple propagation rule.

Apparently, the earliest publications on PDS are Brueni [13], Baldwin et al. [8], and Mili et al. [54]. Later, Haynes et al. [31] showed that the problem is NP-complete even when the input graph is bipartite; they presented a linear time algorithm to solve PDS optimally on trees. Kneis et al. [43] generalized this result to a linear time algorithm that finds an optimal solution for graphs that have bounded tree-width, relying on earlier results of Courcelle et al. [21]. Kneis et al. [43] also showed that PDS is a generalization of the Dominating Set problem as follows. Given a graph G we can construct an augmented graph G' such that S is an optimal solution for the Dominating Set problem on G if and only if it is an optimal solution for PDS on G'; the graph G' is obtained from G by adding a new node v' for each node v in G and adding the edge vv'. Guo et al. [30] developed a combinatorial algorithm based on dynamic programming for optimally solving PDS on graphs of tree-width k. The running time of their algorithm is $O(c^{k^2} \cdot n)$ where c is a constant. Dorfling and Henning computed the power domination number, i.e. the size of an optimal power dominating set, for $n \times m$ grids [24].

Guo et al. also compared the tractability of the DOMINATING SET problem versus PDS on several classes of graphs; that is, they study whether there are classes of graphs where the former problem is in P but the latter one is NP-hard; but they have no result that "separates" the two problems. Even for planar graphs, the DOMINATING SET problem is NP-hard [28], and the same holds for PDS [30]. Liao and Lee [48] proved that PDS on split graphs is NP-complete, and also they presented a polynomial-time algorithm for solving PDS optimally on interval graphs. Brueni and Heath [14] have more results on PDS, especially the NP-completeness of PDS on planar bipartite graphs. To the best of our knowledge, no further results are known on solving the PDS problem, either optimally or approximately.

The main results in this chapter are as follows:

- In Section 3.1, we prove that it is NP-hard to approximate $\gamma(G)$ within a factor of $2^{\log^{1-\epsilon} n}$.
- In Section 3.2, we introduce the notion of strong regions and weak regions as a mean of obtaining lower bounds on $\gamma(G)$. Based on this, we develop an approximation algorithm for PDS that gives an approximation guarantee of $(k+1)$ for graphs that have tree-width k. The algorithm requires the tree decomposition as part of the input, and runs in time $O(n^3)$, independent of k. Our algorithm provides an approximation algorithm with a guarantee of $O(\sqrt{n})$ for PDS on planar graphs because a tree decomposition of a planar graph with width $O(\sqrt{n})$ can be computed efficiently [5]. Moreover, we show that our methods (specifically, the lower bounds used in our analysis) cannot improve on our approximation guarantee of $O(\sqrt{n})$.
- In Section 3.3, we introduce two extensions of the PDS problem, the DIRECTED PDS problem and the ℓ-round PDS problem. We show that our hardness of approximation result for the PDS problem can be carried over to these two problems.

Moreover, in Chapter 4, we present a linear time dynamic programming algorithm for DIRECTED PDS when the underlying undirected graph has bounded tree-width. We also present a polynomial-time dynamic programming algorithm to compute the optimal solution to the ℓ-round PDS problem for weighted graphs with bounded tree-width.

3.1 Hardness of approximation

In this section, we prove that the PDS problem cannot be approximated within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$. In Section 3.4, we improve the complexity assumption, and prove that it is NP-hard to approximate the PDS problem within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$; this result is an easy extension of the weaker result. We prove our weaker result by providing a *gap preserving reduction* from MinRep to PDS.

3.1.1 The reduction from MinRep to PDS

In the MinRep [44] problem we are given a bipartite graph $G = (A, B, E)$ with a partition of A and B into equal-sized subsets. Let $A = A_1 \cup A_2 \cup \cdots \cup A_{q_A}$ denote the partition of A, and let $B = B_1 \cup B_2 \cup \cdots \cup B_{q_B}$ denote the partition of B. The goal in the MinRep problem is to pick a minimum-size set of nodes, $A' \cup B' \subseteq V(G)$, to cover all super edges in $\mathcal{H}$. (Refer to Section 2.3.2 for more discussion on the MinRep problem.)

Theorem 3.1.1 *The PDS problem cannot be approximated within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$.*

The reduction: Theorem 3.1.1 is proved by a reduction from the MinRep problem. We create an instance $\overline{G} = (\overline{V}, \overline{E})$ of the PDS problem from a given instance $G = (A, B, E)(\mathcal{H} = (\mathcal{A}, \mathcal{B}, \mathcal{E}))$ of the MinRep problem. The idea is to replace each super edge with a "cover testing gadget".

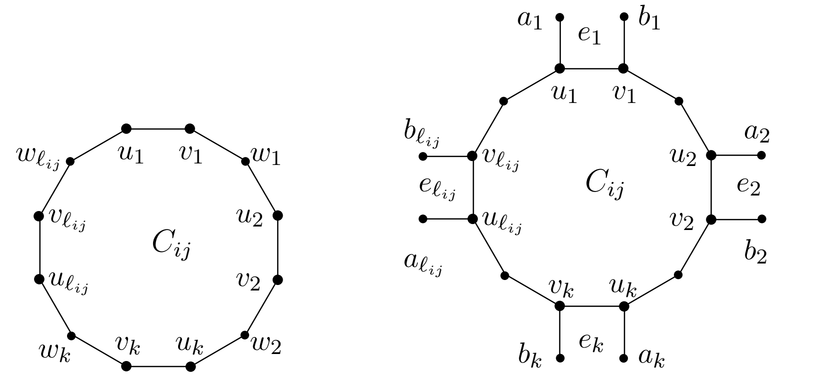
1. Start with a copy of each node in $A \cup B$ in $\overline{G}$. For convenience, we use the same notation for nodes (and set of nodes) in G and their copies in $\overline{G}$.

2. Add a new node w^* to the graph $\overline{G}$, and connect w^* to all nodes in $A \cup B$. Also add three new nodes w_1^*, w_2^*, w_3^* and connect them to w^*; these three nodes force any optimal solution to contain w^*.

3. $\forall i \in \{1, \ldots, q_A\}, j \in \{1, \ldots, q_B\}$ if A_iB_j is a super edge, then do the following:

   (a) Let E_{ij} denote the set of edges between A_i and B_j in G and let ℓ_{ij} denote $|E_{ij}|$ (see Figure 3.2(a); for an example E_{11} has 3 edges, and E_{12} has 4 edges). We denote the edges in E_{ij} by $e_1, e_2, \cdots, e_k, \cdots, e_{\ell_{ij}}$.

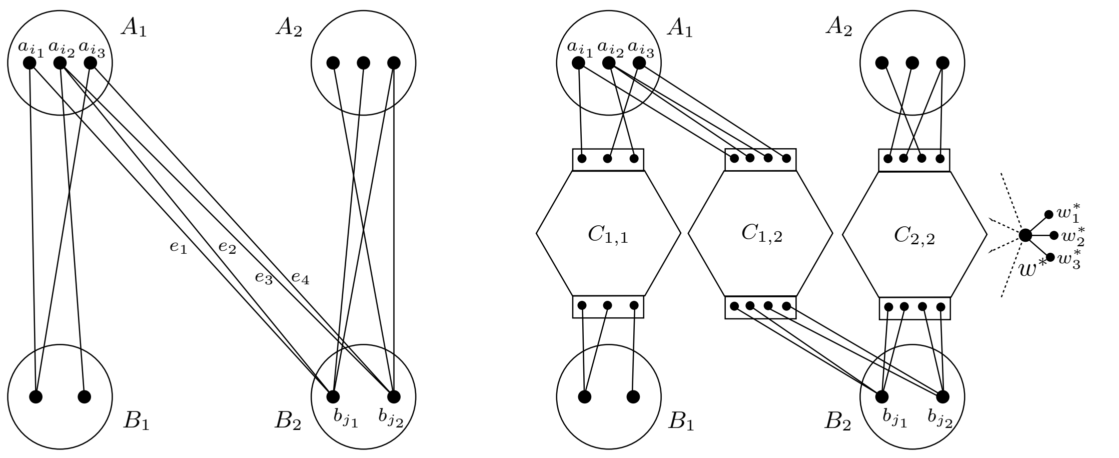
   (b) Let C_{ij} be a cycle of $3\ell_{ij}$ nodes. We sequentially label the nodes of C_{ij} as $u_1, v_1, w_1,\ u_2, v_2, w_2,\ \cdots, u_k, v_k, w_k, \cdots$ (informally speaking, we associate each triple u_k, v_k, w_k with an edge e_k of E_{ij}). Make $\lambda = 4$ new copies of the graph C_{ij} (λ can be any constant greater than 3; refer to the proof of Lemma 3.1.2 for more details). For each edge $e_k = a_kb_k \in E_{ij}$ and for each of the 4 copies of C_{ij}, we add an edge from a_k to u_k and an edge from b_k to v_k. See Figures 3.1(a), 3.1(b) for an illustration.



(a) The C_{ij} graph (b) Edges between C_{ij} and $A_i \cup B_j$.

Figure 3.1: The cover testing gadget.

4. Let $\overline{G} = (\overline{V}, \overline{E})$ be the obtained graph (see Figure 3.2 for an illustration).



(a) MinRep Instance G

(b) PDS instance $\overline{G}$: For each super edge A_iB_j we show only 1 copy of C_{ij}; in fact $\overline{G}$ has $\lambda = 4$ copies of C_{ij}.

Figure 3.2: The hardness construction

Let S be a solution for the resulting PDS instance $\overline{G}$, and suppose $w^* \in S$. Then all of the nodes in $A \cup B$ are power dominated (by rule (R1) of PDS). Now consider a gadget C_{ij}, and assume a node v of C_{ij} is in S. By applying rule (R1) once and then repeatedly applying rule (R2) of PDS, the gadget C_{ij} will be completely power dominated; that is, all nodes of the gadget will be in $\mathcal{P}^*(S)$.

The next lemma shows that the size of an optimal solution in PDS is exactly one more than the size of an optimal solution in MinRep. The number of nodes in the constructed graph is equal to $\left|V(\overline{G})\right| = 4 + |V(G)| + 3\lambda\,|E(G)|$. This will complete the proof of Theorem 3.1.1 by showing that the above reduction is a gap preserving reduction from MinRep to PDS with the same gap (hardness ratio) as the MinRep problem; refer to the Hardness of Approximation chapter in Vazirani's book [72] for the definition of a gap preserving reduction.

Lemma 3.1.2 *$A^* \cup B^*$ is an optimal solution to the instance $G = (A, B, E)$ of the* MinRep *problem if and only if $S^* = A^* \cup B^* \cup \{w^*\} \subseteq V(\overline{G})$ is an optimal solution to the instance $\overline{G}$ of the PDS problem.*

Proof: First, we claim that w^* should be in any optimal solution, S^*, of the PDS instance $\overline{G}$. Suppose that w^* is not in some optimal solutions. Then, in order to power dominate the nodes w^*, w_1^*, w_2^*, w_3^* in $\overline{G}$, the set S^* must contain at least two of the nodes (leaves) w_1^*, w_2^*, w_3^*; since otherwise the two non-picked leaf nodes cannot be covered by applying the propagation rule to w^*. This is a contradiction, since we can replace these 2 nodes by w^* and obtain a smaller solution.

Assume that $A^* \cup B^*$ is a solution for the MinRep instance G. We will show that $S^* = A^* \cup B^* \cup \{w^*\}$ is a solution to the PDS instance $\overline{G}$. Note that all nodes in $A \cup B \cup \{w^*, w_1^*, w_2^*, w_3^*\}$ are power dominated by applying rule (R1) on w^*. Now, we only need to show that all nodes in the gadgets C_{ij} are power dominated. Consider any super edge A_iB_j of $\mathcal{H}$. The set $A^* \cup B^*$ covers all super edges in $\mathcal{H}$. Hence, there exists a pair of nodes $a_k \in A^* \cap A_i$, $b_k \in B^* \cap B_j$ that induces an edge of G. Since a_k and b_k are in S^*, their neighbors, u_k and v_k, in each of the $\lambda = 4$ copies of C_{ij} in $\overline{G}$, will be power dominated by applying rule (R1). Then the nodes u_k and v_k in each copy of C_{ij} will power dominate the entire cycle by repeatedly applying rule (R2). To see this, note that any node in C_{ij} has exactly 2 neighbors in C_{ij} and at most 1 neighbor not in C_{ij}. The neighbors not in C_{ij} are from $A_i \cup B_j$, and they are power dominated by w^*. Hence, if a node in C_{ij} and one of its neighbors in C_{ij} are power dominated, then by applying rule (R2) the other neighbor in C_{ij} will be power dominated. Hence, by starting from v_k and repeatedly applying rule (R2), we can sequentially power dominate the nodes in C_{ij}. This shows that S^* power dominates all nodes in $\overline{G}$. Therefore, $\gamma(\overline{G})$ is at most $|A^* \cup B^*| + 1$.

Let $S^* \subseteq V(\overline{G})$ be an optimal solution for PDS. By the above claim, w^* is in S^*. Now define $A' = A \cap S^*$ and $B' = B \cap S^*$. First we prove that any optimal solution of PDS is contained in $A \cup B \cup \{w^*\}$, and then we show that $A' \cup B'$ covers all super edges of the MinRep instance G. Suppose that S^* contains some nodes not in $A \cup B \cup \{w^*\}$. Hence, there are some gadgets that are not completely power dominated by $S^* \cap (A \cup B \cup \{w^*\})$. Let C_{ij} be such a gadget. By symmetry each of the $\lambda = 4$ copies of C_{ij} is not completely power dominated. Therefore, the optimal solution S^* needs to have at least 3 nodes from the $\lambda = 4$ copies of C_{ij}. By removing these 3 nodes from S^* and adding $a_k \in A_i$ and $b_k \in B_j$ to S^* for some arbitrary edge $a_kb_k \in E_{ij}$, we can power dominate all of the 4 copies of C_{ij}. This contradicts the minimality of S^*, and proves that $S^* \subseteq A \cup B \cup \{w^*\}$. To see that $A' \cup B'$ covers all super edges, note the following: suppose no node from any copy of C_{ij} is in the optimal solution; then any C_{ij} can be power dominated only by taking a pair of nodes $a \in A_i$, $b \in B_j$ that induces an edge of G. This completes the proof of the lemma. □

3.2 Approximation algorithms for planar graphs

In this section, we describe an approximation algorithm with a guarantee of $(k+1)$ for PDS in graphs with tree-width k; the running time is $O(n^3)$, independent of k. This algorithm gives an approximation algorithm with a guarantee of $O(\sqrt{n})$ for PDS in planar graphs. Finally, we show that the analysis of our algorithm is tight on planar graphs. We use Planar PDS to denote the special case of the PDS problem where the graph is planar.

We introduce the notion of a *strong* region before presenting our algorithm. Informally speaking, a set of nodes $R \subseteq V$ is called strong if every solution to the PDS problem has a node of R. Let $G = (V, E)$ denote the graph. The neighborhood of $R \subseteq V$ is defined as $nbr(R) = \{v \in V | \exists uv \in E,\ u \in R,\ v \notin R\}$, and the exterior of R is defined as $ext(R) = nbr(V \setminus R)$, i.e., $ext(R)$ consists of the nodes in R that are adjacent to a node in $V \setminus R$.

Definition 3.2.1 *Given a graph $G = (V, E)$ and a set $S \subseteq V$, the subset $R \subseteq V$ is called an S-*strong region *if $R \not\subseteq \mathcal{P}^*(S \cup nbr(R))$, otherwise, the set R is called an S-*weak region*. The region R is called* minimal S-strong *if it is an S-strong region and $\forall r \in R$, $R - r$ is an S-weak region.*

It is easy to check from the definition that an S-strong region is also an $\emptyset$-strong (or shortly strong) region. Any solution to the PDS problem needs to have at least one node from every strong region.

Lemma 3.2.2 *Let $G = (V, E)$ be a given graph, and let $S \subseteq V$. A subset $R \subseteq V$ is an S-strong region if and only if for every set $S' \subseteq V$ such that $S \cup S'$ is a solution to G, we have $R \cap (S' \setminus S) \neq \emptyset$.*

Proof: It can be seen that the set $S \cup (V \setminus R)$ will power dominate the same set of nodes from R that can be power dominated by $S \cup nbr(R)$; this is valid for any subset $R \subseteq V$. Let R be an S-strong region. By the definition of a strong region we have $R \not\subseteq \mathcal{P}^*(S \cup nbr(R))$. Hence, by the above claim $R \not\subseteq \mathcal{P}^*(S \cup (V \setminus R))$. This shows that every solution $S \cup S'$ needs to have at least one node from R that is not in S.

Now assume that for every solution $S \cup S'$ of G we have $R \cap (S' \setminus S) \neq \emptyset$. Suppose that R is an S-weak region, so we have $R \subseteq \mathcal{P}^*(S \cup nbr(R))$. Let $S' = V \setminus R$. It follows that $S \cup S'$ is a solution, but R has no intersection with $S' \setminus S$. This is a contradiction, so R is an S-strong region. □

Our algorithm makes one level-by-level and bottom-to-top pass over the tree T of the tree decomposition of G and constructs a solution S for PDS (initially, $S = \emptyset$). At each node r_j of T we check whether the union of the bags in the subtree rooted at r_j forms an S-strong region; if yes, then the bag X_{r_j} of r_j is added to S, otherwise S is not updated. The key point in the analysis is to show that $\gamma(G)$ is lower bounded by the number of nodes of T where we updated S. This is done by constructing disjoint strong regions corresponding to the nodes where we updated S.

Algorithm 1 $O(k)$-approximation Algorithm

```
1: A tree decomposition $\langle \{X_i | i \in I\}, T\rangle$ of $G$ is given, where $T$ is rooted at $r$.
2: Let $I_\ell$ be the set of $T$-nodes at distance $\ell$ from the root, and let $d$ be the
   maximum distance from $r$ in $T$.
3: $S \leftarrow \emptyset$
4: for $i = d$ to 0 do
5:   Let $I_i = \{r_1, \ldots, r_{k_i}\}$ and denote by $T_{r_j}$ the subtree in $T$ rooted at $r_j$.
6:   Let $Y_{r_j}$ be the union of bags corresponding to the $T$-nodes in $T_{r_j}$.
7:   for $j = 1$ to $k_i$ do
8:     if $Y_{r_j}$ is an $S$-strong region then
9:       $S \leftarrow S \cup X_{r_j}$, where $X_{r_j}$ is the bag corresponding to $r_j$.
10:    end if
11:  end for
12: end for
13: Output $S_o = S$
```

3.2.1 Analysis of the algorithm

In this subsection, we show that our algorithm has an approximation guarantee of $O(k)$. Let $G = (V, E)$ denote the input graph, and let $S \subseteq V$ be any set of nodes.

Lemma 3.2.3 *Suppose Z is an S-weak region such that $ext(Z) \subseteq S$. Then we have $Z \subseteq \mathcal{P}^*(S)$.*

Proof: Let $Y = ext(Z)$, it follows from the definitions that $nbr(Z \setminus Y) \subseteq ext(Z)$. We claim that $Z \setminus Y$ is an S-weak region. Let $S^* = V \setminus (Z \cup S)$, it is easy to check that $S \cup S^*$ is a solution for the graph G, but $S^* \cap (Z \setminus Y) = \emptyset$. Hence, by Lemma 3.2.2, $Z \setminus Y$ is not an S-strong region, and so it is an S-weak region. Thus, $Z \setminus Y \subseteq \mathcal{P}^*(S \cup nbr(Z \setminus Y)) \subseteq \mathcal{P}^*(S \cup ext(Z)) = \mathcal{P}^*(S)$ and this implies that $Z \subseteq \mathcal{P}^*(S)$ as $Y = ext(Z) \subseteq S$. □

Lemma 3.2.4 *Let $Z \subseteq V$ be an S-strong region. Suppose that Y is a subset of V such that $Y \subseteq \mathcal{P}^*(S)$ and $ext(Y) \subseteq S$. Then $Z \setminus Y$ is an S-strong region.*

Proof: Assume for the sake of contradiction that $Z \setminus Y$ is an S-weak region. Then by the definition of strong regions we have $Z \setminus Y \subseteq \mathcal{P}^*(S \cup nbr(Z \setminus Y))$. It is easy to see that $nbr(Z \setminus Y) \subseteq nbr(Z) \cup ext(Y)$. This implies that $Z \setminus Y \subseteq \mathcal{P}^*(S \cup nbr(Z \setminus Y)) \subseteq \mathcal{P}^*(S \cup nbr(Z) \cup ext(Y)) = \mathcal{P}^*(S \cup nbr(Z))$. The condition in the lemma states that $Y \subseteq \mathcal{P}^*(S) \subseteq \mathcal{P}^*(S \cup nbr(Z))$. Hence, we get $Z = (Z \setminus Y) \cup (Z \cap Y) \subseteq \mathcal{P}^*(S \cup nbr(Z))$, which means that Z is an S-weak region. This is a contradiction, so the lemma is proved. □

Theorem 3.2.5 *Given a graph $G = (V, E)$ and a tree decomposition of G of width k as input, Algorithm 1 runs in time $O(n \cdot |E|)$, and achieves an approximation guarantee of $(k + 1)$.*

Proof: First, we show that the solution S_o found by the algorithm is feasible. Then we prove the approximation guarantee, and establish the running time.

For any node q of T, recall that Y_q denotes the union of the bags corresponding to the T-nodes in the subtree rooted at q in T. We claim that $ext(Y_q) \subseteq X_q$. Suppose that q has m children in T, call them $c_1, \ldots, c_m$. For each edge qc_j $(j = 1, \cdots, m)$, the set $X_q \cap X_{c_j}$ *separates* Y_{c_j} from the rest of the graph; that is, every path between a node in Y_{c_j} and a node in $V \setminus Y_{c_j}$ contains a node of $X_q \cap X_{c_j}$ (see Lemma 12.3.1 in [23]). Thus, $ext(Y_{c_j}) \subseteq X_q \cap X_{c_j} \subseteq X_q$, and hence, for $Y_q = X_q \cup Y_{c_1} \cup \cdots \cup Y_{c_m}$, we have $ext(Y_q) \subseteq X_q$.

We use induction on the height of the subtree of T rooted at q to prove the following: if Y_q is S^*-strong, then $Y_q \subseteq \mathcal{P}^*(S^* \cup X_q)$, where S^* denotes the solution just before the algorithm examines Y_q. The statement clearly holds when q is a leaf of T (since $Y_q = X_q$). Otherwise, let $c_1, \ldots, c_m$ be the children of q in T. For each $j = 1, \ldots, m$, when the algorithm examined Y_{c_j}, either Y_{c_j} was S-weak, in which case (by Lemma 3.2.3) we have $Y_{c_j} \subseteq \mathcal{P}^*(S \cup ext(Y_{c_j})) \subseteq \mathcal{P}^*(S \cup (X_{c_j} \cap X_q)) \subseteq \mathcal{P}^*(S^* \cup X_q)$ or Y_{c_j} was S-strong in which case $Y_{c_j} \subseteq \mathcal{P}^*(S \cup X_{c_j})$ by induction (note that $S \cup X_{c_j} \subseteq S^*$); we use S to denote the solution just before the algorithm examines Y_{c_j}. Hence, $Y_q = Y_{c_1} \cup \cdots \cup Y_{c_m} \cup X_q \subseteq \mathcal{P}^*(S^* \cup X_q)$.

The above statement implies that $V \subseteq \mathcal{P}^*(S_o)$ because at the step when the algorithm examines the root r of T either

(i) Y_r is S-strong, so $S_o = S \cup X_r$, and $Y_r \subseteq \mathcal{P}^*(S \cup X_r) = \mathcal{P}^*(S_o)$; or

(ii) Y_r is S-weak, and $Y_r \subseteq \mathcal{P}^*(S \cup ext(Y_r)) = \mathcal{P}^*(S_o)$; note that $Y_r = V(G)$ and $ext(Y_r) = \emptyset$.

To show that the approximation guarantee is $(k + 1)$ we will construct a set Δ of pairwise disjoint strong regions $R_1, R_2, \ldots$, such that there is a strong region R_j corresponding to each step of the algorithm that adds a non empty bag X_{q_j} to S. Thus, $|S_o| \leq (k + 1)\,|\Delta|$ since each bag has $\leq k + 1$ nodes, and $\gamma(G) \geq |\Delta|$ because every solution has size $\geq |\Delta|$, by Lemma 3.2.2. Hence, $|S_o| \leq (k+1)\gamma(G)$. We construct the sets $R_1, R_2, \ldots$, during the execution of the algorithm as follows. Suppose the algorithm finds Y_q to be S-strong while examining a node q of T. Let $q_1, \ldots, q_{\ell-1}$ be the nodes of T where the algorithm updated the solution before examining q, and let S be the solution just before the algorithm examines q. Then define $R_\ell = Y_{q_\ell} \setminus (Y_{q_1} \cup \cdots \cup Y_{q_{\ell-1}})$, where $q_\ell = q$. We claim that R_ℓ is an S-strong region. For each strong region Y_{q_j} $(j = 1, \ldots, \ell - 1)$ we have seen that $ext(Y_{q_j}) \subseteq X_{q_j} \subseteq S$ and $Y_{q_j} \subseteq \mathcal{P}^*(S)$; note that the algorithm added X_{q_j} to the solution since Y_{q_j} was a strong region. It follows that $ext(Y_{q_1} \cup \cdots \cup Y_{q_{\ell-1}}) \subseteq S$, and $Y_{q_1} \cup \cdots \cup Y_{q_{\ell-1}} \subseteq \mathcal{P}^*(S)$. Hence, by Lemma 3.2.4, the set R_ℓ is an S-strong region. Clearly, the sets $R_1, R_2, \ldots,$ are pairwise disjoint. This completes the construction of Δ.

Consider the running time. Without loss of generality we can assume that the given tree decomposition of width k has at most $4n$ bags (see Lemma 13.1.2 in [42]). Using standard algorithmic techniques we can test in $O(|E|)$ time whether a given set $R \subseteq V$ is an S-strong region (we compute $\mathcal{P}^*(S \cup nbr(R))$ and check if it contains R). Therefore, our algorithm has a running time of $O(n \cdot |E|)$. □

It is known that planar graphs have tree-width $O(\sqrt{n})$ [67], and such a tree decomposition can be found in $O(n^{\frac{3}{2}})$ time [5]. This fact together with the above theorem proves the following theorem.

Theorem 3.2.6 *Algorithm 1 achieves an approximation guarantee of $O(\sqrt{n})$ for the* Planar PDS *problem.*

The dynamic programming algorithm of Guo et al. [30] for the PDS problem finds an optimal solution in linear time on bounded tree-width graphs; the running time of their algorithm is supper polynomial for graphs with $\Omega(\sqrt{\log n})$ tree-width. Our algorithm is based on the notion of strong regions and weak regions, and it finds an approximation solution with approximation guarantee of $\texttt{tw}(G)+1$ in cubic time; the running time is independent of the tree-width.

3.2.2 Lower bounds via disjoint strong regions

In this subsection, we show that any approximation algorithm for PDS that uses the number of disjoint strong regions as a lower bound has an approximation guarantee of $\Omega(\sqrt{n})$.

Lemma 3.2.7 *Any minimal S-strong region is connected.*

Proof: Let G be a given graph and S be a subset of nodes of G. Suppose the lemma is not correct. Let R be a minimal S-strong region which is not connected. Let $C \subset R$ be a connected component of R. The component C is an S-weak region since R is a minimal S-strong region, so by the definition of weak region we have $C \subseteq \mathcal{P}^*(S \cup nbr(C))$. The set C is a maximal connected component of R, so the neighborhood of C has no intersection with $R \setminus C$. This means that $nbr(C) \subseteq nbr(R)$, which implies that $C \subseteq \mathcal{P}^*(S \cup nbr(C)) \subseteq \mathcal{P}^*(S \cup nbr(R))$. The same argument as above shows that $R \setminus C \subseteq \mathcal{P}^*(S \cup nbr(R \setminus C)) \subseteq \mathcal{P}^*(S \cup nbr(R))$. Therefore, $R \subseteq \mathcal{P}^*(S \cup nbr(R))$, which is a contradiction. Hence, the lemma is proved. □

Lemma 3.2.8 *The number of disjoint strong regions in an $\ell \times m$ grid is exactly one.*

Proof: For the sake of contradiction, assume that the given grid has two disjoint strong regions. Take as few nodes as possible from these strong regions until we get minimal strong regions, say R_1 and R_2. It is easy to check that the set of nodes of any row or any column of the grid power dominates all nodes in the grid. By Lemma 3.2.2, R_1 and R_2 should have at least one node from every solution. In the other words, R_1 and R_2 must have at least one node from each row and also from each column. By Lemma 3.2.7 any minimal strong region induces a connected subgraph. Hence, in R_1 there is a path from a node in the top row to a node in the bottom row, and also in R_2 there is a path from a node in the rightmost column to a node in the leftmost column. Obviously these two paths share a common node. This is a contradiction, since R_1 and R_2 are assumed to be disjoint. $\square$

Proposition 3.2.9 (Theorem 1 in [24]) *Let G be an $\ell \times m$ grid with $\ell \leq m$, then $\gamma(G) = \Theta(\ell)$.*

Proof: Let S^* be an optimal solution. From the definition of the PDS problem, it is clear that $X = S^* \cup nbr(S^*)$ is a solution for the PROPAGATION problem. By Corollary 2.2.2, we have $|X| \geq \ell$. The maximum degree in the grid is 4, so we have $\ell \leq |X| \leq 5 \cdot |S^*|$. Therefore, we have $\gamma(G) \geq \frac{\ell}{5}$. $\square$

Consider any approximation algorithm for PDS that uses only the number of disjoint strong regions as a lower bound on the size of an optimal solution. By Lemma 3.2.8, this algorithm finds a lower bound of 1 on the size of an optimal solution on a grid. The $\sqrt{n} \times \sqrt{n}$ grid has an optimal solution of size $\Theta(\sqrt{n})$ by Proposition 3.2.9. This shows that the approximation guarantee of the algorithm is $\Omega(\sqrt{n})$, even on planar graphs.

Proposition 3.2.10 *Consider any approximation algorithm for PDS that uses* only *the number of disjoint strong regions as a lower bound on the optimal value. Then the approximation guarantee is $\Omega(\sqrt{n})$.*

3.3 Extensions of PDS

In this section, we introduce two extensions of the PDS problem, and show that our hardness of approximation result for the PDS problem carry over to these two extensions.

3.3.1 PDS in directed graphs

In this subsection, we extend the PDS problem to directed graphs to obtain the Directed Power Dominating Set (Directed PDS) problem. Our motivation for studying the directed problem comes from theoretical considerations. The Dominating Set problem is studied on both undirected and directed graphs, and there is extensive literature on the latter (see [32, 33]). The similarities between the Dominating Set problem and the PDS problem led us to define and study the Directed PDS problem. We give a result on the hardness of approximation of Directed PDS.

Definition 3.3.1 (the Directed PDS problem) *Let G be a directed graph. Given a set of nodes $S \subseteq V(G)$, the set of nodes that are power dominated by S, denoted by $\mathcal{P}^*(S)$, is obtained as follows:*

(D1) *if node v is in S, then v and all of its out-neighbors are in $\mathcal{P}(S)$;*

(D2) (propagation) *if node v is in $\mathcal{P}(S)$, one of its out-neighbors w is not in $\mathcal{P}(S)$, and all other out-neighbors of v are in $\mathcal{P}(S)$, then w is inserted into $\mathcal{P}(S)$.*

The set $\mathcal{P}(S)$ changes as we apply (R1) and (R2); the final set $\mathcal{P}(S)$ is denoted by $\mathcal{P}^*(S)$. We say that S power dominates G if $\mathcal{P}^*(S) = V(G)$. The Directed PDS *problem is to find a minimum-sized set of nodes S that power dominates all the nodes in G.*

In this section, we prove that the Directed PDS problem cannot be approximated within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$. In Section 3.4, we improve the complexity assumption to $\mathsf{P} \neq \mathsf{NP}$, and we prove that it is NP-hard to approximate the Directed PDS problem within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$. The proof of weaker result uses a reduction from the MinRep problem to the Directed PDS problem in directed *acyclic* graphs. This reduction is similar to the reduction in Theorem 3.1.1; the main difference comes from the gadget for modeling the super edges.

Theorem 3.3.2 *The* Directed PDS *problem even when restricted to directed* acyclic *graphs cannot be approximated within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$.*

The reduction: The reduction is the same as the reduction for the undirected version with minor changes. We are using a different gadget for modeling the super edges. This gadget makes the constructed graph, directed acyclic. In the following we create an instance of Directed PDS, $\overline{G} = (\overline{V}, \overline{E})$, from a given instance $G = (A, B, E)(\mathcal{H} = (\mathcal{A}, \mathcal{B}, \mathcal{E}))$ of the MinRep problem.

1. Add a new node w^* (master node) to the graph G, and add a directed edge from w^* to all nodes in G.

2. $\forall i \in \{1, \ldots, q_A\}, j \in \{1, \ldots, q_B\}$ do the following:

   (a) Let $E_{ij} = \{e_1, e_2, \ldots, e_{\ell_{ij}}\}$ be the set of edges between partition A_i and B_j in G. Remove E_{ij} from G.

   (b) Let D_{ij} be the graph in Figure 3.3 (The incoming dashed line shows an edge from the master node w^* to that node). The D_{ij} gadget has a pair of nodes (u_k, v_k) corresponding to each edge $e_k \in E_{ij}$. Make $\lambda = 4$ new copies of the graph D_{ij} and connect them in the same way to the corresponding super nodes as shown in Figure 3.3. For each k, connect the end nodes of an edge e_k to the corresponding pair of nodes (u_k, v_k) in each copy of D_{ij}; i.e., add directed edges (a_k, u_k) and (b_k, v_k).

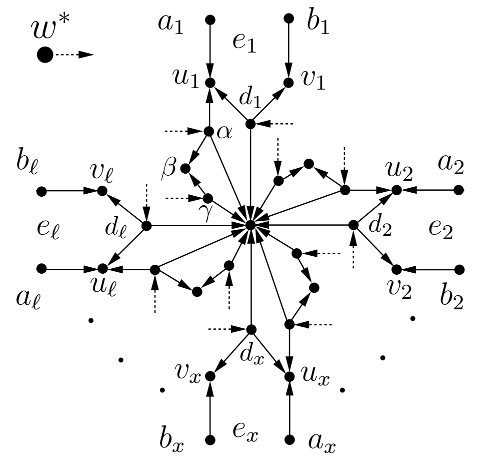


Figure 3.3: The D_{ij} graph and its connection to the super nodes

3. Let $\overline{G} = (\overline{V}, \overline{E})$ be the obtained graph.

The analysis: The next lemma shows that the size of an optimal solution in Directed PDS is exactly one more than the size of an optimal solution in the MinRep instance. The number of nodes in the constructed graph is at most $|V(\overline{G})| \leq 1 + |V(G)| + 7k|E(G)|$. This shows that the above reduction is a *gap preserving reduction* from MinRep to Directed PDS with the same gap (hardness ratio) as the MinRep problem. Therefore, the following lemma will complete the proof of the above theorem. The proof of the following lemma is similar to the proof of Lemma 3.1.2, and we skip it here.

Lemma 3.3.3 (A^*, B^*) *is an optimal solution to the instance $G = (A, B, E)$ of the* MinRep *problem if and only if $S^* = A^* \cup B^* \cup \{w^*\} \subseteq V(\overline{G})$ is an optimal solution to the instance $\overline{G}$ of the* Directed PDS *problem.*

3.3.2 ℓ-round PDS problem

In this subsection, we first introduce a hierarchy of problems between Dominating Set and PDS, by adding a parameter ℓ to PDS which restricts the number of "parallel" rounds of propagation that can be applied (refer to Section 2.4.1 where a similar notion is defined for the Propagation problem). Next, we show that the ℓ-round PDS problem even for $\ell = 4$ is hard to approximate within $2^{\log^{1-\epsilon} n}$. The reduction given here is similar to the reduction used to prove the same hardness of approximation for the Directed PDS problem.

The rules of the ℓ-round PDS problem are the same as PDS, except we try to apply the propagation rule in parallel as much as possible. In the first round we apply the rule (R1) to all the nodes in S, and for the rest of the rounds we only consider "parallel" application of the propagation rule (R2). In every "parallel" round we power dominate all the new nodes that can be power dominated by applying the propagation rule to all of the nodes that are power dominated in the previous "parallel" rounds. Given a parameter ℓ, the ℓ-round PDS problem is the problem in which we want to power dominate all of the nodes in at most ℓ parallel rounds. Now we define the "parallel" propagation rule formally. Given a graph $G = (V, E)$ and a subset of nodes $S \subseteq V$, the set of nodes that can be power dominated by applying at most ℓ rounds of parallel propagation, denoted by $\mathcal{P}^\ell(S)$, is defined recursively as follows:

$$\mathcal{P}^\ell(S) = \begin{cases} \bigcup_{v \in S} N[v] & \ell = 1 \\ \mathcal{P}^{\ell-1}(S) \bigcup \{v : \{u, v\} \in E, N[u] \setminus \{v\} \subseteq \mathcal{P}^{\ell-1}(S)\} & \ell \geq 2 \end{cases}$$

Given a graph $G = (V, E)$ and a parameter ℓ, the goal in the *ℓ-round PDS* problem is to find a minimum size subset of nodes $S \subseteq V$, such that $\mathcal{P}^\ell(S) = V$. Clearly, the ℓ-round PDS problem for $\ell = 1$ is exactly the Dominating Set problem, and for a graph G with n nodes the ℓ-round PDS problem for $\ell \geq n - 1$ is exactly the PDS problem.

A solution for the PDS problem provides a plan for installing monitoring devices to monitor the whole power network, but it does not provide any guarantees on the time-lag between a fault in the network and its detection. Deducing information through a parallel round of propagation takes one unit of time and in some applications we want to detect a failure in the network after at most ℓ units of time. The addition of the parameter ℓ achieves this time constraint.

We first prove the following theorem that gives a result with a weaker complexity assumption of $\mathsf{NP} \not\subseteq \mathsf{DTIME}(n^{polylog(n)})$. In Section 3.4, we improve the complexity assumption to $\mathsf{P} \neq \mathsf{NP}$, and we prove that it is NP-hard to approximate the ℓ-round PDS problem within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$.

Theorem 3.3.4 *The ℓ-round PDS problem for any $\ell \geq 4$ cannot be approximated within $2^{\log^{1-\epsilon} n}$ ratio, for any fixed $\epsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{polylog(n)})$.*

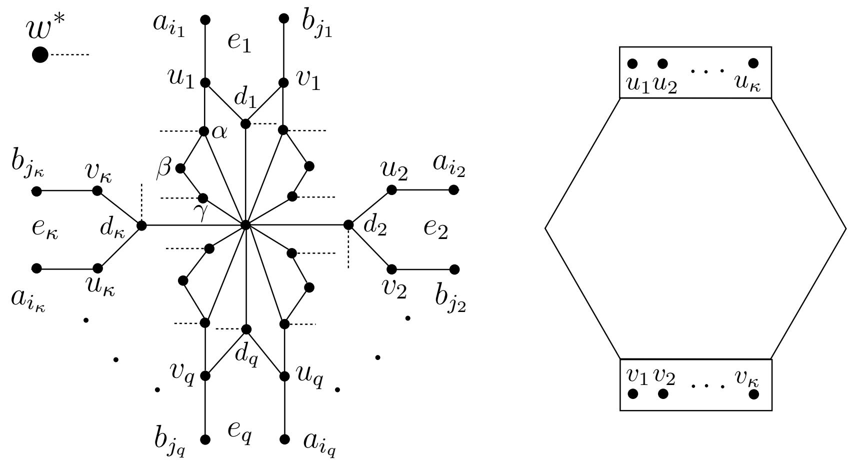


Figure 3.4: The cover testing gadget D_{ij}

The reduction: Theorem 3.3.4 is proved by a reduction from the MinRep problem. In the following we create an instance $\overline{G} = (\overline{V}, \overline{E})$ of the ℓ-round PDS problem from a given instance $G = (A, B, E)(\mathcal{H} = (\mathcal{A}, \mathcal{B}, \mathcal{E}))$ of the MinRep problem.

1. Add a new node w^* (master node) to the graph G, and add an edge between w^* and all the other nodes in G. Also add three new nodes w_1^*, w_2^*, w_3^* and connect them to w^*.
2. $\forall i \in \{1, \dots, q_A\}, j \in \{1, \dots, q_B\}$ do the following:
   (a) Let $E_{ij} = \{e_1, e_2, \dots, e_\kappa\}$ be the set of edges between $A_i = \{a_{i_1}, \dots, a_{i_{m_A}}\}$ and $B_j = \{b_{j_1}, \dots, b_{j_{m_B}}\}$ in G, where κ is the number of edges between A_i and B_j.
   (b) Remove E_{ij} from G.
   (c) Let the edge $e_q \in E_{i,j}$ be incident to a_{i_q} and b_{j_q} (in G). In this labeling for simplicity the same node might get different labels. Let D_{ij} be the graph in Figure 3.4 (a dashed line shows an edge between a node and the master node w^*). Make $\lambda = 4$ new copies of the graph D_{ij} and then identify nodes a_{i_q}'s, b_{j_q}'s with the corresponding nodes in A_i and B_j (in G). Note that the λ copies are sharing the same set of nodes, A_i and B_j, but other nodes are disjoint.
3. Let $\overline{G} = (\overline{V}, \overline{E})$ be the obtained graph.

The analysis: The next lemma shows that the size of an optimal solution in ℓ-round PDS is exactly one more than the size of an optimal solution in the MinRep instance. The number of nodes in the constructed graph is at most $\left|V(\overline{G})\right| \leq 4 + |V(G)| + 10\lambda\,|E(G)|$. This shows that the above reduction is a *gap preserving reduction* from MinRep to ℓ-round PDS with the same gap (hardness ratio) as the

MinRep problem. Therefore, the following lemma will complete the proof of the above theorem. As we mentioned above, the reduction given here is similar to the one used for proving the hardness of the directed PDS problem. One important part of the above construction (see Figure 3.4) is the gadget on the set of nodes $\{\alpha, \beta, \gamma\}$. Note that there should be such a gadget between the center node in $D_{i,j}$ and each u_q and v_q; in Figure 3.4 not all of the gadgets are shown (for example between u_2 and the center node). This gadget introduces direction into undirected construction, it allows the propagation in only one direction. After the center node of $D_{i,j}$ is power dominated, all of the other nodes in $D_{i,j}$ are power dominated (by the propagation rule). On the other hand, the power domination cannot propagate through the gadget in the other direction (toward the center node).

Lemma 3.3.5 *The pair (A^*, B^*) is an optimal solution to the instance $G = (A, B, E)$ of the* MinRep *problem if and only if $\Pi^* = A^* \cup B^* \cup \{w^*\} \subseteq V(\overline{G})$ is an optimal solution to the instance $\overline{G}$ of ℓ-round PDS (for all $\ell \geq 4$).*

Proof: The proof is similar to Lemma 3.1.2. The key property of the gadget is as follows. Let $e_q = \{a_{i_q}, b_{j_q}\}$ be an edge between A_i and B_j. By picking a_{i_q} and b_{j_q}, it is easy to check that all $\lambda = 4$ copies of D_{ij} can be power dominated in 4 parallel rounds. □

3.4 Improved hardness results

In this section, we present our improved hardness result for the PDS, ℓ-round PDS, and Directed PDS problems. We prove our improved hardness results by giving reductions from the Extended Label Cover problem.

Theorem 3.4.1 *For any fixed $\epsilon > 0$, it is NP-hard to approximate the ℓ-round* PDS *problem, for any $\ell \geq 4$, within ratio $2^{\log^{1-\epsilon} n}$.*

Let $\mathcal{C} = (\Psi, X, F, D, R)$ be an instance of the Extended Label Cover problem; refer to Section 2.3.2 for the definition. Recall that each variable $x \in X$ is assigned an integer weight of $\mathtt{W}(x)$; let $\overline{W} = \mathtt{W}(X)$.

Reduction: The reduction is an extension of the reduction used in the proof of Theorem 3.3.4.

1. Start from an empty graph H. Corresponding to each variable x and each $a \in F$ add $\mathtt{W}(x)$ variable nodes, $v^1(x, a), \ldots v^{\mathtt{W}(x)}(x, 1)$, to H. Corresponding to each constraint $\psi \in \Psi$ add $\lambda = \overline{W} + 2$ constraint nodes, $d^1(\psi), \ldots, d^\lambda(\psi)$, to H.

2. Add a new node w^* to H, and connect w^* to all of the variable nodes. Also add new nodes w_1^*, w_2^*, w_3^* and connect them to w^*.

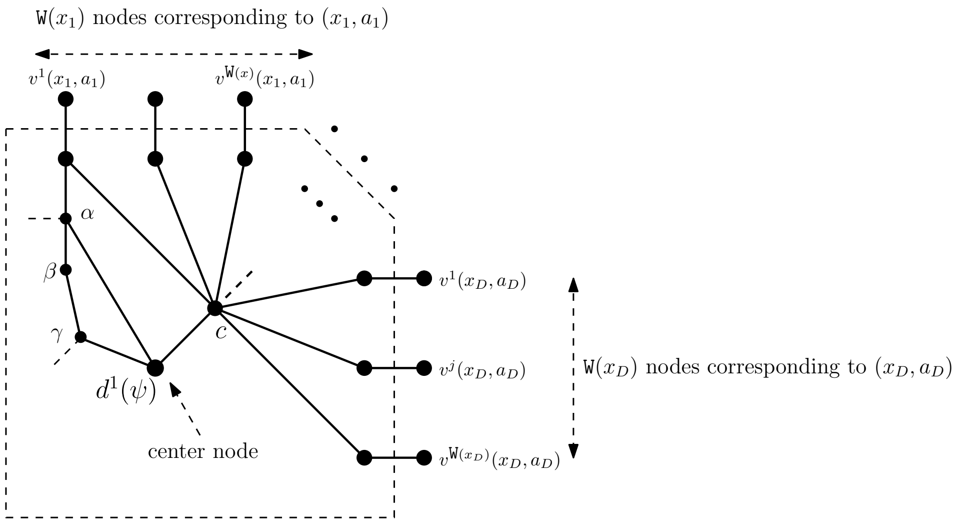


Figure 3.5: First copy of the cover testing gadget corresponding to constraint ψ and satisfying assignment $(a_1, \ldots, a_D) \in R_\psi$.

3. In the next step we "install" copies of the gadget shown in Figure 3.5. The nodes inside the dashed box are called *internal* nodes, and the other nodes are called *external* nodes of the gadget. A dashed edge shows an edge between a node and the master node w^*. An important part of this gadget is the subgraph on the set of nodes $\{\alpha, \beta, \gamma\}$. Note that there should be such a subgraph between the center node and the neighbor of each external node in this gadget; in Figure 3.5 only one of these subgraphs is shown.

4. For each constraint $\psi \in \Psi$ of $\mathcal{C}$ and for each satisfying assignment $(a_1, \ldots, a_D)$ of ψ do the following:
   (a) Assume constraint ψ depends on variables $x_1, \ldots, x_D$.
   (b) Add a copy of the gadget shown in Figure 3.5 between all the copies of the variable nodes corresponding to $(x_1, a_1), \ldots, (x_D, a_D)$ and the ith copy of the center node, $d^i(\psi)$, for each $i = 1, \ldots, \lambda$.

5. Let H be the obtained graph.

The following lemma completes the proof of Theorem 3.4.1. The proof is similar to Lemma 3.1.2, and we skip it here.

Lemma 3.4.2 *If the extended label cover instance $\mathcal{C}$ has an optimal solution of weight W^*, then $\gamma(H) = W^*$. Moreover, $\overline{f}$ is an optimal solution to $\mathcal{C}$ if and only if $\{w^*\} \cup \left\{v^i(x, a) | i = 1, \ldots, \mathtt{W}(x), a \in \overline{f}(x)\right\}$ is an optimal solution to H.*

By ignoring the parameter ℓ in the analysis of the above construction, we get the improved hardness result for the PDS problem.

Theorem 3.4.3 *For any fixed $\epsilon > 0$, it is* NP*-hard to approximate the unweighted* PDS *problem within ratio $2^{\log^{1-\epsilon} n}$.*

Using a directed version of the gadget in Figure 3.5, we can prove the following improved result for the Directed PDS problem.

Theorem 3.4.4 *For any fixed $\epsilon > 0$, it is* NP*-hard to approximate the* Directed PDS *problem within ratio $2^{\log^{1-\epsilon} n}$ even when restricted to directed acyclic graphs.*

3.5 Conclusion

We studied the PDS problem from the perspective of approximation algorithms. We introduced a natural extension of the problem to directed graphs, and we also introduced the ℓ-round PDS problem. We showed that these problems have a threshold of $O(2^{\log^{1-\epsilon} n})$ for the hardness of approximation. We presented an approximation algorithm with a guarantee of $O(\sqrt{n})$ for Planar PDS.

Here, we describe an algorithm with an approximation guarantee of $O(\frac{n}{\log n})$ for the PDS problem. The algorithm works as follows. Partition the nodes of the graph G into $\log n$ equal-sized sets $V_1, V_2, \cdots$. Next, consider all possible subfamilies of these sets, and consider the union of sets in each subfamily as a candidate solution for the PDS problem on G. Among all these different candidates, output the one that power dominates G and has the minimum number of nodes. Note that in the algorithm we only consider $2^{\log n} = n$ different candidates. Clearly, the algorithm runs in polynomial time, since the feasibility of each candidate can be tested in polynomial time. Let S^* be an optimal solution. It is easy to see that the set of V_i's that intersect S^* forms a solution for the PDS problem in G; this solution has size at most $\frac{n}{\log n} \cdot |S^*|$. This establishes the approximation guarantee. The same algorithm and analysis applies to the Directed PDS and ℓ-round PDS problems.

Proposition 3.5.1 *There is a polynomial-time approximation algorithm with a guarantee of $\frac{n}{\log n}$ for the PDS,* Directed PDS, *and ℓ-round PDS problems.*

There is a gap between our hardness threshold of $O(2^{\log n^{1-\epsilon}})$ and our approximation guarantee of $O(\frac{n}{\log n})$, and narrowing this gap is an open question.

A major open question in the area is whether there exists a PTAS (*polynomial time approximation scheme*) for Planar PDS. A first step may be to obtain an improvement on our approximation guarantee of $O(\sqrt{n})$. There has been a lot of research on designing PTASs for NP-hard problems on planar graphs. Some of the most important developments are the outerplanar layering technique by Baker [7], and the bidimensionality theory by Demaine and Hajiaghayi [22]. Unfortunately, these methods do not apply to Planar PDS; this is explained in the following two paragraphs.

Baker [7] showed that the Dominating Set problem in planar graphs has a PTAS. In the Baker method we first partition the graph into smaller graphs. Then we solve the problem optimally on each subgraph, and finally we return the union of the solutions as a solution for the original graph. The example in Figure 2.1 shows that this method does not apply to Planar PDS. The size of an optimal solution is 1, but if we apply the Baker method, then the size of the output solution will be at least as large as the number of subgraphs in the partition which can be $\Theta(n)$.

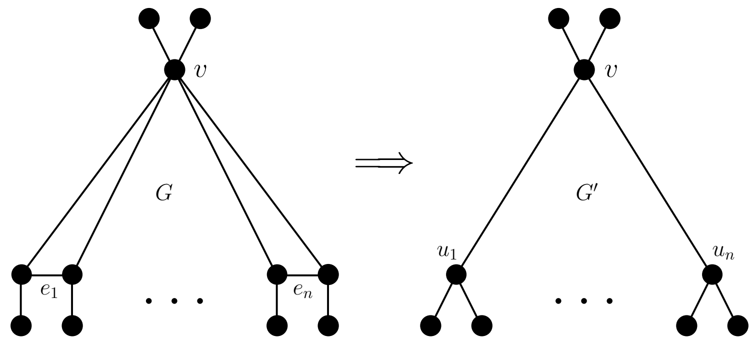


Figure 3.6: Optimal value of PDS increases when edges are contracted.

Demaine and Hajiaghayi [22] introduced the bidimensionality theory and used it to obtain PTASs for several variants of the Dominating Set problem on planar graphs. An important property of bidimensionality is that when an edge is contracted the size of an optimal solution should not increase. Consider the example in Figure 3.6. If we contract edges $e_1, e_2, \ldots, e_n$ in G, then we get the graph G'. It can be checked that $\gamma(G) = 1$, but $\gamma(G') = \Theta(n)$. Thus, the bidimensionality theory does not apply to Planar PDS since the optimum value may increase when an edge is contracted.

Chapter 4

Algorithms for the general propagation problem

In this chapter, we present dynamic programming algorithms for the PDS problem (defined in Chapter 2) and the Propagation problem (defined in Chapter 3). We introduce a common generalization of both problems called the General Propagation problem. We reformulate it as an orientation problem. Our dynamic programming algorithm is based on this reformulation, and it can optimally solve the generalized problem on graphs with bounded tree-width. We introduce a restricted version of the problem called the ℓ-round General Propagation problem. We present a PTAS for this problem on planar graphs for small values of the parameter ℓ.

Courcelle et al. [21] proved that, on graphs of bounded tree-width, any problem that can be formulated in *monadic second-order logic* is solvable in linear time. Kneis et al. [43] formulated the PDS problem in the monadic second-order logic; this gives a linear time algorithm for PDS on graphs of bounded tree-width. Guo et al. [30] developed a combinatorial algorithm based on dynamic programming for optimally solving PDS on graphs with bounded tree-width in linear time. The key idea in their combinatorial algorithm is a new formulation of PDS in terms of "valid orientation" of edges. Our dynamic programming algorithms are built on this orientation formulation.

The outerplanar layering technique by Baker [7], and the bidimensionality theory by Demaine and Hajiaghayi [22] are two important developments on planar graphs (refer to Section 3.5 for more discussion). Baker [7] showed that the Dominating Set problem in planar graphs has a PTAS. Demaine and Hajiaghayi [22] introduced the bidimensionality theory and used it to obtain PTASs for several variants of the Dominating Set problem on planar graphs.

The main results in this chapter are as follows:

- In Section 4.1, we design a linear time dynamic programming algorithm to optimally solve the General Propagation problem in weighted graphs of bounded tree-width. Next, we present a polynomial-time dynamic programming algorithm to optimally solve the ℓ-round General Propagation problem in bounded tree-width graphs. Our methods are based on extending the orientation formulation for the PDS problem, given by Guo et al. [30], to the General Propagation and ℓ-round General Propagation problems.

- In Section 4.2, we present a PTAS for the ℓ-round General Propagation problem on planar graphs for small values of ℓ. Our methods are based on Baker's methods [7], and it turns out that Baker's PTAS for the Dominating Set problem is a special case of our result.
- In Section 4.3, we focus on directed graphs, and we give a linear time algorithm based on dynamic programming for directed PDS when the underlying undirected graph has bounded tree-width.
- In Section 4.4, we focus on the Target Set Selection problem from social network theory. We show that our algorithmic results from Sections 4.1 and 4.2 can be adapted to provide similar results for this problem.

4.1 Dynamic programming for bounded tree-width graphs

4.1.1 Reformulation of the General Propagation problem

In this subsection we introduce the General Propagation problem, a problem that contains both the PDS and Propagation problems as special cases. The General Propagation problem is reformulated as an orientation problem; this reformulation is an extension of the orientation formulation for the PDS problem given by Guo et al. [30]. Based on this reformulation a linear time dynamic programming algorithm is provided for bounded tree-width graphs.

Let $G = (V, E)$ be an undirected graph. In the General Propagation problem, there are two types of nodes: *domination* and *simple* nodes. The set of domination nodes are denoted by V_D, and the rest of the nodes (i.e., $V \setminus V_D$) are simple nodes. The set V_D is part of the input, so in each instance of the General Propagation problem the type of each node is fixed. There are two covering rules. Informally speaking, in the first rule a domination node covers all of its neighbors in addition to itself, but a simple node covers only itself. The second rule is the same as the propagation rule stated in the PDS and Propagation problems. The formal definitions are given below. Given a set of nodes S the set of nodes *covered* by S, denoted $\mathcal{P}^*(S)$, is obtained as follows.

(R1) (a) if $v \in S$ is a *domination* node, then v and all of its neighbors are added to $\mathcal{P}(S)$;

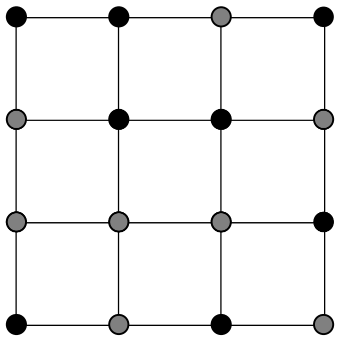


Figure 4.1: An instance of the General Propagation problem

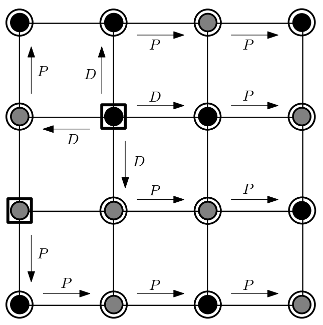
(b) if $v \in S$ is a *simple* node, then v is added to $\mathcal{P}(S)$

(R2) (*propagation*) if node u is in $\mathcal{P}(S)$, one of its neighbors, v, is not in $\mathcal{P}(S)$, and all other neighbors of u are in $\mathcal{P}(S)$, then v is inserted into $\mathcal{P}(S)$.

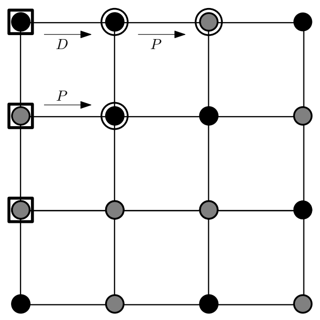
The set $\mathcal{P}(S)$ changes as we apply (R1) and (R2); the final set $\mathcal{P}(S)$ is denoted by $\mathcal{P}^*(S)$. We say that node v is picked if v is in the set S, and we say that v is covered if it is inserted into $\mathcal{P}(S)$ by either rule (R1) or (R2). We denote the size of the optimal solution that covers all nodes of G by $\delta(G, V_D)$ (or, for short, by $\delta(G)$). When we apply rule (R1) to domination node u to cover node v, we say that v is covered by applying the *domination* rule to u and denote it by $u \xrightarrow{D} v$. Similarly, when we apply rule (R2) to node u to cover node v, we say that v is covered by applying the *propagation* rule to u and denote it by $u \xrightarrow{P} v$. When all nodes are of the domination type (i.e., $V_D = V$) we get the PDS problem, and when all nodes are of simple type (i.e., $V_D = \emptyset$) we get the Propagation problem.

Consider the 4×4 grid shown in Figure 4.1 and assume that the black nodes are the domination nodes and the gray nodes are the simple nodes. Consider Figure 4.2(a) and assume that we have picked the two nodes denoted by a square. First, by applying the domination rule to the black node, we cover all four of its neighbors; these dominations are denoted by the 4 arrows labeled by D in the figure. Next, by applying the propagation rule sequentially to the nodes we can cover all the remaining nodes in this graph. The applications of the propagation rule are denoted by arrows labeled by P. The covered nodes are enclosed in circles. Figure 4.2(b) shows the same 4×4 grid, where we have picked different nodes. The three nodes picked cannot cover the whole graph. We can only apply the domination rule to the first node in the first row to cover its neighbor in the second column. We can apply the propagation rule only twice, as shown in the figure. At this stage, any covered node either has no non-covered neighbors or has at least 2 non-covered neighbors. Therefore, we cannot apply the propagation rule any more. The covered nodes are enclosed in circles.

We reformulate the General Propagation problem as an orientation problem; our dynamic programming algorithms are based on the new formulation. Infor-



(a) Feasible solution

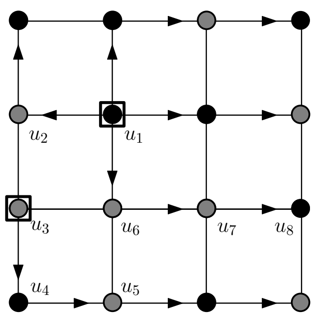


(b) Infeasible solution

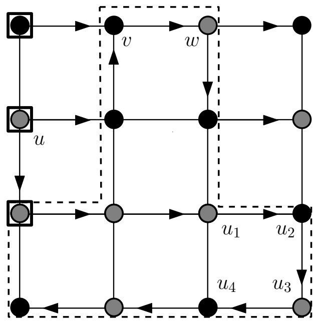
Figure 4.2: Applications of covering rules; picked nodes are enclosed in a square, and covered nodes are enclosed in a circle.

mally speaking, the orientation of edges shows how the propagation rule is applied; that is, we orient the edge $\{u, v\}$ from u to v if $u \overset{D/P}{\rightarrow} v$.

Definition 4.1.1 *An* orientation *of an undirected graph $G = (V, E)$ is obtained by assigning an orientation to some (but not necessary all) edges of G. We denote an orientation of G by $\widehat{\mathcal{O}} = (V, E_d, E_u)$ where E_d is the set of oriented edges and E_u is the set of undirected edges.*



(a) Valid orientation



(b) Invalid orientation

Figure 4.3: Orientations of the 4×4 grid given in Figure 4.1.

Definition 4.1.2 *A* dependency path, *$P = v_0, e_1, v_1, e_2, v_2, \ldots,$ in an orientation $\widehat{\mathcal{O}}$ is a sequence of some edges such that P has no two consecutive unoriented edges and each oriented edge e_i in P is oriented from v_{i-1} to v_i (i.e., all oriented edges are directed away from the start node of P). The length of a dependency path is defined as the number of oriented edges in the path. A* dependency cycle *is a dependency path that starts and ends at the same node.*

Figure 4.3 shows two different orientations for the example of the General Propagation problem given in Figure 4.1. Consider the first orientation given in 4.3(a). The path $P = u_1, u_2, u_3, u_4, u_5, u_6$ is a dependency path since all oriented edges are in the same direction and there are no two consecutive unoriented edges. The path $P' = u_1, u_2, u_3, u_6, u_5$ is not a dependency path since it has two consecutive unoriented edges. Also, the path $P'' = u_1, u_6, u_5, u_4$ is not a dependency path, since (u_5, u_4) is a backward edge. It can be checked that there are no dependency cycles in this orientation.

Definition 4.1.3 *A* valid orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ *of an undirected graph* $G = (V, E)$ *is an orientation of* G *with the following properties:*

(O1) *Each node of the directed graph* $G_d = (V, E_d)$ *has in-degree at most* 1*:*

$$\forall v \in V : d^-_{G_d}(v) \leq 1$$

(O2) *A* simple *node with no in-coming edges has at most* 1 *out-going edge:*

$$\forall v \in V \setminus V_D : d^-_{G_d}(v) = 0 \Longrightarrow d^+_{G_d}(v) \leq 1$$

(O3) *A node with an in-coming edge has at most* 1 *out-going edge;*

$$\forall v \in G : d^-_{G_d}(v) = 1 \Longrightarrow d^+_{G_d}(v) \leq 1$$

(O4) *G has no* dependency cycle.

A node with in-degree 0 *in* $G_d = (V, E_d)$ *is called a* source *of* $\widehat{\mathcal{O}}$.

Consider the graph in Figure 4.1 and its orientation given in 4.3(a). It can be checked that there are no dependency cycles in this orientation, and also all of the degree constraints (O1-O3) are satisfied. Hence, the considered orientation is a valid orientation. Now consider the second orientation given in Figure 4.3(b). This orientation is not valid since it violates the properties of a valid orientation:

- The condition (O1) is violated because the node v has in-degree 2
- The condition (O2) is violated because the simple node u has in-degree 0 and out-degree 2
- The condition (O3) is violated because w has in-degree 1 but out-degree 2,
- The condition (O4) is violated because the cycle $C = u_1, u_2, u_3, u_4, u_1$ is a dependency cycle; note there are no 2 consecutive unoriented edges, and all edges are in the same direction. There are several other dependency cycles, for example, there is a dependency cycle of length 9 containing the edge (v, w), indicated by dashed lines.

Theorem 4.1.4 *Let G be an instance of the* General Propagation *problem. The graph G has a valid orientation with S as the set of sources if and only if $\mathcal{P}(S) = V(G)$.*

Proof: Let V_D be the set of nodes of domination type. Suppose $S \subseteq V(G)$ is a solution to the General Propagation problem; thus, $\mathcal{P}^*(S) = V(G)$. Then we give a valid orientation $\widehat{\mathcal{O}}$ with S as the set of sources by orienting the edges in G according to the way that S propagates in G. We orient an edge (v, w) from v toward w if either v is a domination node and w is covered by applying the domination rule to v (i.e., $v \overset{D}{\to} w$), or node w is covered by applying the propagation rule to v (i.e., $v \overset{P}{\to} w$). When we apply the covering rules to S, we first apply all possible domination rules, and only after that we apply the propagation rules. Also, we do not apply the domination rule or the propagation rule to cover previously covered nodes. It is easy to check that with this orientation the degree requirements (O1-O3) are satisfied; each node can be covered at most once (O1), the propagation rule can be applied to a node v and cover at most one of the neighbors of v (O2-O3).

Now, we need to prove that there is no dependency cycle. We write $v < w$ when a node w is covered in a round after node v. By way of contradiction, suppose that $C^* = u_1, u_2, \ldots, u_m$ is a dependency cycle. Note that all oriented edges in C^* are in the same direction, say oriented in the forward direction. First we claim that there is no source node of domination type in C^*. Suppose u_2 is a source node (i.e., $u_2 \in S$) of domination type in C^* such that the covering rule (R1) is applied to u_2 before any other node in this cycle, and assume that $\{u_2, u_3\}$ is oriented from u_2 to u_3. Since u_2 is a source node, then either $\{u_1, u_2\}$ is oriented in the backward direction (i.e., from u_2 to u_1) or it is unoriented. The first case is not possible, since in C^* all oriented edges are in the same direction. If $\{u_1, u_2\}$ is unoriented, then $\{u_m, u_1\}$ cannot be oriented from u_m to u_1; note that the covering rule is applied to u_2 before any other node in C^*, so u_1 cannot be covered through u_m. Hence, the two consecutive edges incident to u_1 in C^* are unoriented. This is a contradiction, so C^* has no source node of domination type.

Now, focus on the edges of C^*. Assume that all edges in C^* are oriented. Then the oriented edges (u_i, u_{i+1}) imply that $u_i < u_{i+1}$ for all $i = 1, 2, \ldots, m-1$; therefore, $u_1 < u_2 < \cdots < u_m$, but this is a contradiction since the last oriented edge from u_m back to u_1 implies that $u_m < u_1$. Hence, there is no dependency cycle with all edges oriented. Now, assume that the dependency cycle C^* has some unoriented edges. We show that a similar contradiction occurs when there are no two consecutive unoriented edges. Consider an unoriented edge $\{v, w\}$ of C^*, and consider the maximum sequence $w_1 = w, \ldots, w_\ell$ of oriented edges starting from w in the cycle C^*; such a sequence exists since there are no two consecutive unoriented edges in C^*. Note that there is an in-coming edge to v in the cycle C^*, so v cannot be a source node. Therefore, by the propagation rule, we see that v should be covered before the propagation rule can be applied to $w_1 = w$ to cover w_2; thus, we have $w_2 > v$. The oriented edge from w_i to w_{i+1} (for $i \geq 2$) shows that $w_{i+1} > w_i$.

Combining these dependencies, we see that $v < w_\ell$; i.e., v is covered before the last node, in the sequence of oriented edges starting from w_1, is covered. Repeating this argument, we get an ordering in which some of the nodes in C^* are covered. This cyclic ordering gives a contradiction. As an example, suppose that m is even and the edges of C^* are alternatively unoriented and oriented (starting with an unoriented edge); then we get $u_1 < u_3 < u_5 < \cdots < u_{m-1} < u_1$ (see Figure 4.4 for an example) and this gives the contradiction $u_1 < u_1$. Hence, the first part of the theorem follows: If S is a solution to the General Propagation problem, then G has a valid orientation with S as the set of sources.

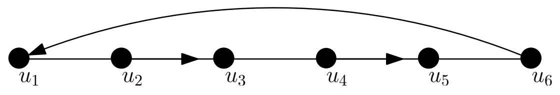


Figure 4.4: A dependency cycle

Now suppose that G has a valid orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ with $S \subseteq V(G)$ as the set of sources. Note that the domination nodes in S (i.e., $S \cap V_D$) cover all their out-neighbors in the orientation $\widehat{\mathcal{O}}$; the set of nodes covered by these applications of (R1) is $Y = S \cup \{w \in V : \exists v \in S \cap V_D, (v, w) \in E_d\}$. If there is a node v in Y with an out-neighbor $w \notin Y$ such that v has no other neighbors in $V \setminus Y$, then by applying rule (R2) we can cover w; note that we applied the propagation rule (R2) to v to cover w according to the orientation $\widehat{\mathcal{O}}$. Now, we prove that by applying the propagation rule to nodes in Y (according to the orientation $\widehat{\mathcal{O}}$) we can cover all nodes in G. Suppose that this does not occur; that is, $\mathcal{P}^*(S) \neq V(G)$. Let $X \subset V$ be the maximal set of nodes that can be covered by Y; that is, $\mathcal{P}^*(S) = X$. We claim that there is at least one oriented edge from X to $V \setminus X$. Note that all of the sources are in X, so each of the nodes in $V \setminus X$ has in-degree 1 in $G_d = (V, E_d)$. Hence, if there is no oriented edge from X to $V \setminus X$, then there should be an oriented cycle in $G[V \setminus X]$. This is not possible since there are no dependency cycles. Therefore, there is at least one oriented edge from X to $V \setminus X$. Let $e_1 = (x_1, y_1), \ldots, e_k = (x_k, y_k)$ be all of the oriented edges from X to $V \setminus X$. If some x_i has all of its neighbors in X except y_i, then by applying the propagation rule to x_i the node y_i will be covered. By the maximality assumption of X this cannot happen. Therefore, each x_i has another neighbor, say z_i, in $V \setminus X$. Then (x_i, z_i) is an unoriented edge by the out-degree requirement; either x_i has in-degree 1 and (O2) applies or x_i is a simple node and (O3) applies, so x_i has out-degree at most 1. Now, we construct a dependency cycle as follows: starting from x_1, use an unoriented edge to move to a node z_1 in $V \setminus X$; then move in the reverse direction over a sequence of oriented edges $(z_2, z_1), (z_3, z_2), \cdots$ (such a sequence exists since each $z_i \in V \setminus X$ is not a source node, so it has in-degree 1) until we reach an oriented edge (z_i, z_{i-1}) with $z_{i-1} \in V \setminus X$ and $z_i \in X$ (such an edge exists since $G[V \setminus X]$ has no oriented cycle); note that (z_i, z_{i-1}) is one of the oriented edges $(x_1, y_1), \cdots, (x_k, y_k)$. If $z_i = x_1$, then we have a dependency cycle; otherwise, we again use an unoriented edge (z_i, z_{i+1}) to move to a node in $V \setminus X$. By repeating these steps, we will eventually find a dependency cycle. Note that all oriented edges are in the same direction and there are no two consecutive unoriented edges in this cycle. This is a contradiction, since $\widehat{\mathcal{O}}$ has no dependency cycle. Hence, we have $X = V$, so S covers G. Hence, the second part of the theorem follows: If G has a valid orientation with S as the set of sources, then $\mathcal{P}^*(S) = V(G)$. □

Theorem 4.1.5 *Let $(G, \mathtt{W})$ be a node-weighted graph that has bounded tree-width; that is, $\mathtt{tw}(G) = O(1)$ and $\mathtt{W}\colon V(G) \to \mathbb{R}$. There is a linear time dynamic programming algorithm that solves the* General Propagation *problem on $(G, \mathtt{W})$.*

The proof is given in Appendix B, which shows the design and analysis of a dynamic programming algorithm; the dynamic program makes critical use of the tree decomposition of G of bounded tree-width.

4.1.2 Reformulation of the ℓ-round General Propagation problem

In this subsection, we introduce an extension of the General Propagation problem called the ℓ-round General Propagation problem. We reformulate it as an orientation problem, and based on this reformulation, we provide a polynomial-time dynamic programming algorithm for bounded tree-width graphs.

Let us formalize the notion of parallel rounds. Let $G = (V, E)$ be a given graph, and assume that V_D is the set of nodes of domination type. Given a positive integer ℓ, and a subset of nodes $S \subseteq V$, the set of nodes that can be covered by applying, at most, ℓ rounds of parallel propagation, denoted by $\mathcal{P}^\ell(S, V_D)$, is defined recursively as follows. To simplify the notation, when the set V_D is clear from the context, we denote the set of covered nodes by $\mathcal{P}^\ell(S)$.

$$\mathcal{P}^\ell(S) = \begin{cases} S & \ell = 0 \\ S \bigcup N[S \cap V_D] \bigcup \{v : \{u,v\} \in E, N[u] \setminus \{v\} \subseteq S\} & \text{if } \ell = 1 \\ \mathcal{P}^{\ell-1}(S) \bigcup \{v : \{u,v\} \in E, N[u] \setminus \{v\} \subseteq \mathcal{P}^{\ell-1}(S)\} & \text{if } \ell > 1 \end{cases}$$

In the first round, we apply the domination rule to all of the domination nodes in S and cover $N[S \cap V_D]$, and we apply one parallel round of the propagation rule to the simple nodes in S and cover $\{v : \{u,v\} \in E, N[u] \setminus \{v\} \subseteq S\}$. In more detail, if u is a domination node, then all of its neighbors are added to $\mathcal{P}^1(S)$ by the domination rule. If u is a simple node, and all of its neighbors except v are in S, then v is added to $\mathcal{P}^1(S)$ by the propagation rule. After the first round, there is no distinction between the domination nodes and the simple nodes, and we apply the propagation rule in $\ell - 1$ parallel rounds.

The goal in the ℓ-round General Propagation problem is to find a minimum-sized subset of nodes $S \subseteq V$, such that $\mathcal{P}^\ell(S) = V$. We use $\delta_\ell(G, V_D)$ to denote the size of an optimal solution for the ℓ-round General Propagation problem on G, where V_D is the set of domination nodes. An extension of the problem is also of interest: the goal is to cover a given subset V' of the nodes in G – i.e., find a minimum-sized set of nodes S such that $V' \subseteq \mathcal{P}^{\ell}(S)$. We denote an instance of the extended problem by $\langle G, V_D, V' \rangle$, and we denote by $\langle G, \mathtt{W}, V_D, V' \rangle$ an instance of the problem when G is a weighted graph with node-weights $\mathtt{W}$. We reformulate the ℓ-round General Propagation problem as an orientation problem. In order to make sure that nodes are covered in at most ℓ parallel rounds, we need to know the round in which nodes are covered in addition to the way that the propagation occurs.

Definition 4.1.6 (valid timed-orientation) *Let $\langle G, V_D, V' \rangle$ be an instance of the extended ℓ-round* General Propagation *problem. A* valid-timed orientation *for $\langle G, V_D, V' \rangle$ is an* orientation *$\widehat{\mathcal{O}} = (V, E_d, E_u)$ (see Definition 4.1.1) together with a time vector $\{t_v : v \in V\}$ (where $t_v \in \{0, 1, \dots, \ell\} \cup \{+\infty\}$) that satisfies the following properties:*

(P1) $\forall v \in V' : 0 \leq t_v \leq \ell,$

(P2) $\forall v \in V : 1 \leq t_v \leq \ell \Rightarrow d^-_{G_d}(v) = 1,$

(P3) $\forall v \in V : t_v = +\infty \Rightarrow d^-_{G_d}(v) = d^+_{G_d}(v) = 0,$

(P4) $\forall v \in V : t_v = 0 \Rightarrow d^-_{G_d}(v) = 0,$

(P5) $$\forall (u, v) \in E_d : t_v = \begin{cases} 1 & \text{if } t_u = 0 \text{ and } u \in V_D \\ 1 + \max\{t_w : w \in N[u] \setminus v\} & \text{otherwise.} \end{cases}$$

where $G_d = (V, E_d)$. A node v with $t_v = 0$ is called a source of the valid timed-orientation.

Consider the 4×4 grid in Figure 4.5, and suppose that the black nodes are of the domination type and the gray nodes are of the simple type. Assume that the goal is to cover all nodes (i.e., $V' = V$) in $\ell = 5$ parallel rounds. Picking the nodes enclosed in a square, we can cover all nodes in 5 parallel rounds. The label assigned to each node shows the round in which the node is covered. It is easy to check that the orientation shown, together with the time-label assigned to the nodes, satisfy properties (P1-P5); moreover, the orientation and the labels correspond to the way that the parallel propagation rule is applied. For example, focus on the property (P5) and consider the following two oriented edges, $e_1 = (u, v)$, and $e_2 = (u', v')$. Domination node u is picked, so it covers all of its neighbors (e.g., v) in 1 round. Simple node u' is picked, but it has 3 neighbors that are not picked. Thus, u' cannot cover v' unless all of its neighbors except v' are covered, and this happens in the 2nd parallel round.

Theorem 4.1.7 *Let $\langle G, V_D, V' \rangle$ be an instance of the extended ℓ-round* General Propagation *problem. Then $S \subseteq V$ is the set of sources of a valid ℓ-round orientation if and only if $V' \subseteq \mathcal{P}^{\ell}(S, V_D)$.*

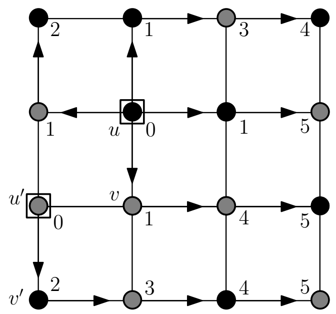


Figure 4.5: A valid timed-orientation of a 4×4 grid

Proof: Suppose $S \subseteq V$ covers V' in at most ℓ parallel rounds; e.g., $V' \subseteq \mathcal{P}^{\ell}(S, V_D)$. We construct a valid ℓ-round orientation with S as the set of sources as follows. We orient the edges in the same way as the domination rule and the propagation rule apply. Consider an edge $\{u, v\} \in E(G)$ and assume that v is covered by applying either the propagation rule or the domination rule to u, then we orient the edge $\{u, v\}$ from u toward v. We do not apply any one of the rules to cover a previously covered nodes; this is needed to make sure that the property (P2) is satisfied. Let E_d be the set of oriented edges and let E_u be the rest of the edges. This defines an orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ of G. Now we define the time vector. For any node v that is not covered let $t_v = +\infty$, and for any other node $v \in \mathcal{P}^{\ell}(S, V_D)$ let $t_v = \min\{r : r \geq 0, v \in \mathcal{P}^r(S)\}$. It is straightforward to check that the above orientation and time vector $\{t_v : v \in V\}$ satisfies properties (P1)-(P5) in the definition of the valid ℓ-round orientation. Hence, there is a valid ℓ-round orientation with S as the set of sources.

Now assume that G has a valid ℓ-round orientation with S as the set of sources. Consider the corresponding orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ and the time vector $(t_v : v \in V)$ that satisfied properties (P1)-(P5). Define $V_r = \{v \in V : 0 \leq t_v \leq r\}$. We claim that $V_r \subseteq \mathcal{P}^r(S, V_D)$. First note that this completes the proof, since any node $v \in V'$ has $0 \leq t_v \leq \ell$ by (P1). We prove the claim by induction on r. For the base case $r = 1$ we need to check that $V_1 \subseteq S \cup N[S \cap V_D] \cup \{v : (u, v) \in E, N[u] \setminus \{v\} \subseteq S\} = \mathcal{P}^1(S, V_D)$. By (P5), each node $v \in V_1$ with $t_v = 1$ has an in-coming edge from a source node u with $t_u = 0$. There are two cases to consider:

1. u is a domination node: The node v can be covered by the domination rule in the first round; i.e., $v \in N[S \cap V_D]$.
2. u is a simple node: so by (P5) all neighbors of u except v have time-label 0. Hence, $N[u] \setminus v \subseteq S$, and consequently v can be covered in the first round by applying the propagation rule to u; i.e., $v \in \{v : (u, v) \in E, N[u] \setminus \{v\} \subseteq S\}$

Now assume that the induction hypothesis holds for $r = k$ (where $k \geq 1$), $V_k \subseteq \mathcal{P}^k(S, V_D)$. Consider a node v such that $t_v = k + 1$. By (P2) it has exactly one incoming edge, say $(u, v) \in E_d$. The node u has $t_u \geq 1$, so by property (P5) any $w \in N[u] - v$ has $t_w \leq k$. Therefore, $N[u] \setminus \{v\} \subseteq \mathcal{P}^k(S, V_D)$. This is correct for any node v with $t_v = k + 1$, so $X = \{v \in V : t_v = k + 1\} \subseteq \{v : (u, v) \in E, N[u] \setminus \{v\} \subseteq \mathcal{P}^k(S, V_D)\}$. Hence, by the definition of the parallel propagation rule X can be covered in one parallel round, so we have $V_{k+1} = V_k \cup X \subseteq \mathcal{P}^{k+1}(S, V_D)$. This proves the induction step and completes the proof. □

Theorem 4.1.8 *Given an instance $\langle G, \mathtt{W}, V_D, V' \rangle$ of the ℓ-round* General Propagation *problem where $G = (V, E)$ is an undirected graph with tree-width k, a minimum weight set $S \subseteq V$ such that $V' \subseteq \mathcal{P}^\ell(S)$ can be obtained in time $O(c^{k^2 + k \log \ell} \cdot |V|)$, for a constant c. Moreover, if G is a planar graph, then the algorithm has the running time of $O(c^{k \log \ell} \cdot |V|)$.*

The proof is given in Appendix B, which shows the design and analysis of a dynamic programming algorithm.

4.2 PTAS for ℓ-round General Propagation problem on planar graphs

In this section, we provide a PTAS for the ℓ- round General Propagation problem on planar graphs for $\ell = O(\frac{\log n}{\log \log n})$.

Consider a fixed embedding of a planar graph G. We define the nodes at level i denoted by L_i as follows [7]. Let L_1 be the set of nodes on the exterior face of the given embedding of G. For $i > 1$, the set L_i is defined as the set of nodes on the exterior face of the graph induced on $V \setminus \cup_{j=1}^{i-1} L_j$; that is, we delete the nodes in $L_1 \cup \ldots L_{i-1}$ from the embedding of G, then take L_i to be the set of nodes on the exterior face. We denote by $L[a, b] = \cup_{i=a}^{b} L_i$ the set of nodes at levels a through b. A planar graph is called *d-outerplanar* if it has an embedding where no node is at a level greater than d. For example, consider the graph in Figure 4.6. Clearly, the graph is a 2-outerplanar graph. The set $L_1 = \{u_1, u_2, \ldots, u_8\}$ is the set of nodes at level 1, and the set $L_2 = \{v_1, v_2, \ldots, v_8\}$ is the set of nodes at level 2.

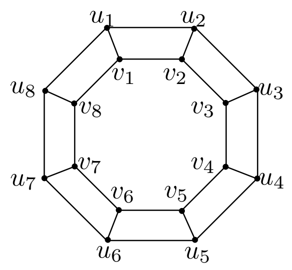


Figure 4.6: A 2-outerplanar graph

Before describing our PTAS, let us look at Baker's PTAS [7] for the Dominating Set problem on planar graphs. Given the parameter $\epsilon = \frac{1}{k}$, Baker's algorithm finds a solution with a size within $(1+\epsilon)$ times the optimal value. This algorithm considers k different decompositions, $\mathcal{D}_1, \ldots, \mathcal{D}_k$, of the nodes of G and then finds a solution for each of them. The ith decomposition consists of blocks with $k+1$ consecutive levels. The jth block in decomposition $\mathcal{D}_i$ contains nodes of levels $jk+i$ through $(j+1)k+i$. (Note that each $\mathcal{D}_i$ is obtained from $\mathcal{D}_1$ by shifting the levels.) Also note that every two consecutive blocks in a given decomposition share a common level. Next, the algorithm solves the Dominating Set problem optimally on each block of $\mathcal{D}_i$. This is possible since each block is $(k+1)$-outerplanar; hence, it has a tree-width of at most $3(k+1)-1$ [11], and the Dominating Set problem can be solved optimally by dynamic programming [7]. (The result in Section 4.1.2 for $\ell = 1$ also shows this fact). It then takes the union of the optimal solutions for the blocks in $\mathcal{D}_i$ to obtain a solution for the Dominating Set problem in graph G. Let S_i denote this solution. Thus, we get k solutions $S_1, \ldots, S_k$ from the decompositions $\mathcal{D}_1, \ldots, \mathcal{D}_k$. Finally, the algorithm outputs the solution that has minimum size, i.e., $\min_{i=1,\ldots,k} |S_i|$, among all k decompositions. It is not hard to argue that the size of this solution is within $\frac{k+1}{k} = 1+\epsilon$ times the optimal value. The key property needed for this argument is the fact that consecutive blocks share a common level.

Algorithm 2 PTAS for ℓ-round General Propagation

```
1: Given a planar embedding of $G$, and the parameter $0 < \epsilon \leq 1$.
2: Let $k = 4 \cdot \lceil \frac{\ell}{\epsilon} \rceil$.
3: for $i = 1$ to $k$ do
4:    for all $j \geq 0$ do
5:       Solve the extended problem on $\mathcal{I}_{i,j} = \langle G[B_{i,j}], V_D \cap C_{i,j}, C_{i,j} \rangle$
6:       Let $\mathcal{O}_{i,j}$ be an optimal solution for $\mathcal{I}_{i,j}$
7:    end for
8:    $\Pi_i = \cup_{j \geq 0} \mathcal{O}_{i,j}$
9: end for
10: $r \leftarrow argmin \{\mathtt{W}(\Pi_i) : i = 1, \cdots, k\}$
11: Output $\Pi_O = \Pi_r$.
```

Let us describe our PTAS informally. Consider a parameter k, that is a function of the parameter ℓ and the approximation factor $(1+\epsilon)$; k will be defined later in the formal description of our algorithm. Consider a fixed planar embedding of G. We decompose the graph in k different ways, $\mathcal{D}_1, \ldots, \mathcal{D}_k$. In each decomposition the graph is decomposed into blocks of $k+4\ell$ consecutive levels. The jth block in $\mathcal{D}_i$ is defined as $B_{i,j} = L[jk+i-2\ell, (j+1)k+i-1+2\ell]$. We denote the k middle levels of $B_{i,j}$ by $C_{i,j} = L[jk+i, (j+1)k+i-1]$. In our PTAS, for each decomposition $\mathcal{D}_i$, we optimally solve instances $\mathcal{I}_{i,j} = \langle G[B_{i,j}], V_D \cap C_{i,j}, C_{i,j} \rangle$ of the extended ℓ-round General Propagation problem in each block of $\mathcal{D}_i$. Note that each instance $\mathcal{I}_{i,j}$ can be optimally solved by using the dynamic programming algorithm given in Section 4.1.2. Let $\mathcal{O}_{i,j}$ denote the optimal solution for this instance. We then take the union of the solutions corresponding to blocks in $\mathcal{D}_i$; $\Pi_i = \cup_{j\geq 0}\mathcal{O}_{i,j}$. By doing this for all k decompositions we get k different solutions for the original graph G. Finally, we choose the solution with minimum weight among these k solutions. We will see that the 2ℓ extra levels around the k middle levels plays an important role in the feasibility and the near optimality of the final output of the algorithm.

To make the role of common levels clear, consider an instance of the 4-round General Propagation problem shown in Figure 4.6. Suppose that all nodes in this graph are of domination type (i.e., $V_D = V$). Hence, the considered problem is actually the 4-round PDS problem. Assume that we partition the planar graph into two blocks. The first block is the outer cycle and the second block is the inner cycle. It is easy to check that the size of an optimal solution in any one of these blocks is 1. For example $\{u_1\}$ and $\{v_5\}$ are optimal solutions for the first block and the second block respectively. But if we consider the original graph G, it is straightforward to check that $S = \{u_1\} \cup \{v_5\}$ is not a solution; the set of nodes that can be power dominated in at most four parallel rounds is $\mathcal{P}^4(S) = \{u_1, u_2, u_5, u_8, v_1, v_4, v_5, v_6\}$. Note that the propagation rule was applied in the subgraph but not in the original graph, so in the subgraph u may have all but one node of $N[u]$ in $\mathcal{P}(S)$ but this need not hold for the original graph (see u_8, for example).

To prevent such difficulties we need to consider extra levels around each block as we did in the blocks $B_{i,j}$ above. We will show that our algorithm finds a solution S for the instance $\mathcal{I}_{i,j}$ such that S in the original graph G covers at least the k middle levels of $B_{i,j}$. This implies that the union of the solutions for the blocks in a given decomposition will be a solution for G, since the union of the k middle levels of the blocks covers all the nodes in G.

Theorem 4.2.1 *Let ℓ be a given parameter, where $\ell = O(\frac{\log n}{\log\log n})$. Then Algorithm 2 is a PTAS for the weighted ℓ-round* General Propagation *problem on planar graphs.*

Proof: Let Π^* be an optimal solution for the ℓ-round General Propagation problem in G. To prove the theorem, it is enough to prove the following two claims:

1. $\mathcal{O}_{i,j}$ is a solution for the instance $\langle G, V_D, C_{i,j}\rangle$ of the extended ℓ-round General Propagation problem,

2. $\Pi^* \cap B_{i,j}$ is a solution for $\langle G[B_{i,j}], V_D \cap C_{i,j}, C_{i,j}\rangle$.

First let us see how the theorem follows from the above claims. The first claim shows that Π_i, for each i, is a solution for the ℓ-round General Propagation problem on G, since $\cup_{j\geq 0} C_{i,j} = V$. The second claim shows that $\mathtt{W}(\mathcal{O}_{i,j}) \leq \mathtt{W}(\Pi^* \cap B_{i,j})$, so $\mathtt{W}(\Pi_i) \leq \sum_j \mathtt{W}(\Pi^* \cap B_{i,j})$. In the right hand side we have counted the nodes in the optimal solution twice on 4ℓ common levels between any two consecutive blocks. By considering all values of the parameter i, $1 \leq i \leq k$, we can find i such that the weight of double counted nodes from the optimal solution Π^* is at most $\frac{4\ell}{k}\mathtt{W}(\Pi^*)$.

This implies that $\mathtt{W}(\Pi_O) \leq (1 + \frac{4\ell}{k})\mathtt{W}(\Pi^*)$, where Π_O is the output of the algorithm and $\mathtt{W}(\Pi_O) = \min_{1\ldots k} \mathtt{W}(\Pi_i)$. By setting $k = 4 \cdot \lceil \frac{\ell}{\epsilon} \rceil$, we get an $(1+\epsilon)$ approximation algorithm.

Now we analyze the running time of the algorithm. In step (5) we solve an instance of the extended ℓ-round General Propagation problem. The graph in this instance has $k + 4\ell$ levels, so it is $(k + 4\ell)$-outerplanar. It is known that the tree-width of any d-outerplanar graph is at most $3d - 1$ [11]. Therefore, step (5) of our algorithm can be done in $c^{O(\ell/\epsilon \log \ell)}$ time by Theorem 4.1.8, since the considered graph has tree-width at most $3(k+4\ell) - 1 < 12(\lceil \frac{\ell}{\epsilon} \rceil + \ell)$. This shows that ℓ should be $O(\log n / \log \log n)$ in order to have a polynomial-time algorithm and in this case step (5) can be done in $n^{O(\frac{1}{\epsilon})}$ time. Also note that the value of j can be at most $\frac{n}{k}$, since the number of levels in a planar graph is at most n (the number of nodes) and each $C_{i,j}$ has k levels. This shows that the algorithm executes step 5 at most $k \times \frac{n}{k} = n$ times. Therefore, the running time of the algorithm is polynomial in the number of nodes of G for a fix $\epsilon > 0$.

Proof of the first claim: We know that $\mathcal{O}_{i,j}$ is a solution for $\langle G[B_{i,j}], V_D \cap C_{i,j}, C_{i,j} \rangle$. Let t_v denotes the round in which node v is covered in $G[B_{i,j}]$. Hence, any node $v \in C_{i,j}$ and possibly some nodes $v \in B_{i,j} \setminus C_{i,j}$ satisfy $0 \leq t_v \leq \ell$. For simplicity we use $L(s)$ to denote the levels $L[jk+i-2(\ell-s), (j+1)k+i-1+2(\ell-s)]$ for any $s \geq 0$; note that $L(0) = B_{i,j}$ and $L(\ell) = C_{i,j}$. Observe that $L(s+1)$ (for $s \geq 1$) is obtained from $L(s)$ by deleting the first two and the last two levels of $L(s)$.

First note that by taking $\mathcal{O}_{i,j}$ all nodes with $t_v = 0$ in $L(0)$ are covered in G. Next, we claim the following: for each s, $0 \leq s \leq \ell$, all nodes $v \in L(s)$ with $t_v \leq s$ can be covered in G in at most s parallel rounds. We prove the statement by induction on s. The base case $s = 0$ is trivial. Assume that the statement is correct for all $s < s'$. Consider a node v with $t_v = s'$ which lies in $L(s')$, and assume that it was covered by applying the propagation rule to node u (see Figure 4.7 for an illustration). It is easy to see that u is inside of the levels

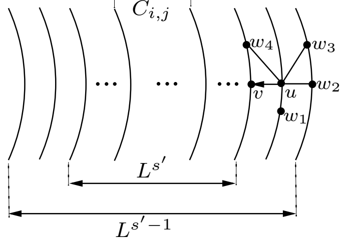


Figure 4.7: Illustration for the induction step

$L[jk+i-2(\ell-s')-1, (j+1)k+i-1+2(\ell-s')+1]$, and therefore all of the neighbors of u are inside the levels $L[jk+i-2(\ell-s')-2, (j+1)k+i-1+2(\ell-s')+2] = L(s'-1)$. The node v was covered by u, so any $w \in N[u] - v$ satisfies $t_w \leq s' - 1$. Then by the induction hypothesis any node $w \in N[u] - v$ is already covered in G, so v can be covered by applying the propagation rule to u. The case when a node is covered by applying a domination rule is similar. This completes the induction step and proves the statement. An important property is that all nodes with $t_v = s'$ should be covered in parallel. Note that by the induction hypothesis nodes with $t_v < s'$ are all covered in the parallel round $s' - 1$, so all nodes with $t_v = s'$ can be covered in parallel at the parallel round s'. To prove the first claim it is enough to note that $L(\ell) = C_{i,j}$, so all nodes in $C_{i,j}$ with $0 \leq t_v \leq \ell$ (that is, all the nodes in $C_{i,j}$) can be covered in G in ℓ parallel rounds.

Proof of the second claim: We know that Π^* is a solution for G. Let t_v denotes the round in which $v \in B_{i,j}$ is covered in G. Hence, each $v \in B_{i,j}$ satisfies: $0 \leq t_v \leq \ell$. Define $L(s)$ as before. The same induction hypothesis as above proves the following: for each s, $0 \leq s \leq \ell$, all nodes v with $t_v \leq s$ in $L(s)$ can be covered in the induced subgraph $G[B_{i,j}]$ staring with $\Pi^* \cap B_{i,j}$. The proof is similar to the proof of the first claim. Note that the set $L(\ell)$ is $C_{i,j}$, so all of the nodes in $C_{i,j}$ can be covered in $G[B_{i,j}]$ in at most ℓ parallel rounds.

In other words, for any solution S^* for G, the intersection with $B_{i,j}$ suffices to cover all of the nodes in $C_{i,j}$ in the subinstance $\langle G[B_{i,j}, V_D \cap C_{i,j}, C_{i,j} \rangle$, but nodes in $B_{i,j} \setminus C_{i,j}$ may not be covered. □

4.3 Directed PDS

We reformulate Directed PDS in terms of *valid colorings* of the edges in order to develop an algorithm based on dynamic programming for Directed PDS. Our method applies to directed graphs such that the underlying undirected graph has bounded tree-width.

There are several notions for the tree-width of directed graphs such as DAG width [60], directed tree-width [38], and Kelly-width [35]. Directed acyclic graphs have width equal to zero for the first two notions [60], and have Kelly-width of 1. Hence, Theorem 3.3.2 gives a hardness threshold of $O(2^{\log n^{1-\epsilon}})$ even if the directed graph has width ≤ 1 according to any of the above three notions.

Definition 4.3.1 *A* coloring *of a directed graph $G = (V, E)$ is a partitioning of the edges in G into red and blue edges. We denote a coloring by $\mathcal{C} = (V, E_r \cup E_b)$ where E_r is the set of red edges and E_b is the set of blue edges.*

We reformulate the Directed PDS problem via a so-called valid coloring of directed graphs; informally speaking, these colorings "model" the application of rules (D1) and (D2) of Directed PDS (see Definition 3.3.1)

Definition 4.3.2 *A* valid coloring *$\mathcal{C} = (V, E_r, E_b)$ of a directed graph $G = (V, E)$ is a coloring of G with the following properties:*

1. No two antiparallel edges can be colored red.

2. The subgraph induced by the red edges, $G_r = (V, E_r)$, has the following properties:

   (a) $\forall v \in G : d^-_{G_r}(v) \leq 1$, and

   (b) $\forall v \in G : d^-_{G_r}(v) = 1 \Longrightarrow d^+_{G_r}(v) \leq 1$.

3. G has no dependency cycle*. A dependency cycle is a sequence of directed edges whose underlying undirected graph forms a cycle such that all the red edges are in one direction, all the blue edges are in the other direction, and there are no two consecutive blue edges.*

We call a node a source *of $\mathcal{C}$ if it has no incoming edges in G_r.*

Our dynamic programming algorithm for Directed PDS is based on the following lemma. The proof of the following theorem is similar to Theorem 4.1.4. Note the similarity between the valid orientation and the valid coloring. Blue edges play the same role as unoriented edges, and Red edges play the same role as oriented edges.

Theorem 4.3.3 *Given a directed graph G and $S \subseteq V(G)$, S power dominates G if and only if there is a valid coloring of G with S as the set of origins.*

Theorem 4.3.4 *Given a directed graph G and a tree decomposition of width k of its underlying undirected graph,* Directed PDS *can be optimally solved in $O(c^{k^2} \cdot n)$ time for a constant c.*

4.4 Target set selection

In this section, we present our algorithmic results for the Target Set Selection problem. Let $G = (V, E)$ be an undirected graph, and assume a *threshold* $t(v)$, where $1 \leq t(v) \leq d(v)$, is assigned to each node $v \in V$. In the Target Set Selection problem, the goal is to select a minimum-sized set of nodes that covers all nodes. The set of nodes that can be covered by S, denoted by $\mathcal{P}^*(S)$, is defined by the following rules.

(R1) Start with $\mathcal{P}(S) = S$ and sequentially apply rule (R2)

(R2) Add node v to $\mathcal{P}(S)$ if at least $t(v)$ of its neighbors are in $\mathcal{P}(S)$.

The set $\mathcal{P}(S)$ changes as we repeatedly apply rule (R2). Informally speaking, first we activate all nodes in S. Next, in each round we activate node v if at least $t(v)$ of its neighbors are active. This process will continue until no new nodes can be activated. Let $\mathcal{P}^*(S)$ denote the final set $\mathcal{P}(S)$. A set S is called a *target set* if $\mathcal{P}^*(S) = V$. The goal in this problem is to find a minimum-sized target set. Given a weight function $\mathtt{W} : V \to \mathbb{Q}$, defined on the nodes of G, we can also ask for the target set S of minimum weight.

The *threshold model* has been introduced to study phenomena in social networks. In this model, each node has a threshold and if the number of its infected neighbors is at least equal to the threshold, then the node becomes infected. In the Target Set Selection problem the goal is to influence the network by targeting a minimum number of nodes. Refer to [40, 41] for further discussion on this model and other related models. These models are closely related to our *propagation model*, although neither model is a special case of the other one. Chen [16] showed that the Target Set Selection problem cannot be approximated within the ratio $2^{\log^{1-\epsilon} n}$, by a reduction from the MinRep problem.

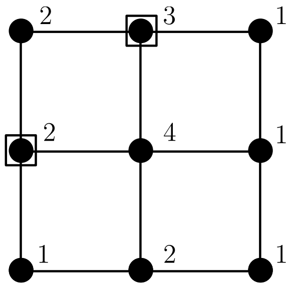
Definition 4.4.1 *A* valid target-orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ *of an undirected graph* $G = (V, E)$ *is an orientation of* G *with the following properties:*

1. *Each node* v *of the graph* $G_d = (V, E_d)$ *has either no in-coming edges or at least* $t(v)$ *in-coming edges;*

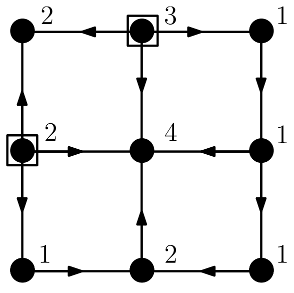
$$\forall v \in G : \; d^-_{G_d}(v) = 0 \text{ or } d^-_{G_d}(v) \geq t(v)$$

2. G_d *has no* directed cycle.

A node with in-degree 0 *in* $G_d = (V, E_d)$ *is called a* source *of* $\widehat{\mathcal{O}}$.



(a) 3×3 grid



(b) Valid orientation

Figure 4.8: Example for the Target Set Selection problem.

Theorem 4.4.2 *Given a graph* $G = (V, E)$ *and the threshold function* $t : V \to \mathbb{Z}^+$, G *has a valid target-orientation with* S *as the set of sources if and only if* S *is a target set for* G.

Proof: Assume S is a target set for G. We orient edges in the following way. Consider an edge $\{u, v\}$. We orient it from u to v if u is covered before v. If both u and v are picked (i.e., $u, v \in S$), then we do not orient the edge $\{u, v\}$. The nodes from S have in-degree 0 in this orientation. A node not in S has at least $t(v)$ in-coming edges, since it has at least $t(v)$ neighbors that are covered before it. It is easy to see that there is no directed cycle in this orientation. Consider the order in which nodes are covered, and note that an edge is oriented from a node, say u, in this ordering toward a node, v, that appears (not necessary immediately) after u.

Assume that there is a valid target-orientation for G with source nodes S. The graph $G_d = (V, E_d)$ is an acyclic digraph. Suppose $u_1, u_2, \dots$ is a topological ordering of G_d. We claim that nodes of G can be covered in this order by activating the nodes in S. To prove the claim, note that all in-neighbors of a node u_i appear before it in the given topological order. Hence, an induction proof on i (the index of nodes in this ordering) proves the claim. □

Consider the 3×3 grid shown in Figure 4.8(a), and assume that the labels assigned to them denote their thresholds. It can be checked that the boxed nodes cover the whole graph. The valid target-orientation corresponding to this solution is shown in Figure 4.8(b); note that the in-degree of each non-picked node is at least its threshold, and also there is no directed cycle in the shown orientation.

Theorem 4.4.3 *There is a polynomial-time dynamic programming algorithm that solves the weighted* Target Set Selection *problem on G with bounded-tree-width. Moreover, the running time of this algorithm is $O(c^{k^2} \cdot T^{k+1} \cdot |V(G)|)$, where T is the maximum threshold of nodes, k is the tree-width of the input graph, and c is a constant.*

4.4.1 PTAS for the ℓ-round Target Set Selection problem

In this section, we introduce the ℓ-round extension of the Target Set Selection problem, and present a PTAS for this problem on planar graphs for small values of ℓ.

Let $G = (V, E)$ be an undirected graph, and assume that $t : V \to \mathbb{N}$ is the threshold function. Given a subset of nodes $S \subseteq V$, the set of nodes that can be activated in at most ℓ parallel rounds, denoted by $\mathcal{P}^\ell(S)$, is defined recursively as follows:

$$\mathcal{P}^\ell(S) = \begin{cases} S & \ell = 0 \\ \mathcal{P}^{\ell-1}(S) \bigcup \left\{v : \left|N(v) \cap \mathcal{P}^{\ell-1}(S)\right| \geq t(v)\right\} & \ell \geq 1 \end{cases}$$

Given a graph $G = (V, E)$ and a parameter ℓ, the goal in the *ℓ-round* Target Set Selection problem is to find a minimum size subset of nodes $S \subseteq V$, such that $\mathcal{P}^\ell(S) = V$. For our PTAS, we also need to solve an extension of the problem in which the goal is to cover a subset V' of nodes. Let $\langle G, V', t\rangle$ denote an instance of the extended problem. Now, we reformulate the ℓ-round Target Set Selection problem as an orientation problem. Informally speaking, an oriented edge shows that the head of the oriented edge is covered after the tail node. We also assign a time label to each node to show the parallel round in which the node is covered.

Definition 4.4.4 (valid ℓ-round target-orientation) *Let $\langle G, V', t\rangle$ be an instance of the extended ℓ-round* Target Set Selection *problem. A* valid ℓ-round target-orientation *for $\langle G, V', t\rangle$ is an* orientation $\widehat{\mathcal{O}} = (V, E_d, E_u)$ *(see Definition 4.1.1) together with a time vector $\{r_v : v \in V\}$ (where $r_v \in \{0, 1, \dots, \ell\} \cup \{+\infty\}$) that satisfies the following properties:*

(P1) $\forall v \in V' : 0 \le r_v \le \ell,$

(P2) $\forall v \in V : r_v = 0 \Rightarrow d^-_{G_d}(v) = 0,$

(P3) $\forall v \in V : r_v = +\infty \Rightarrow d^-_{G_d}(v) = d^+_{G_d}(v) = 0,$

(P4) $\forall v \in V : d^-_{G_d}(v) = 0 \vee d^-_{G_d}(v) \ge t(v),$

(P5) $\forall v \in V : d^-_{G_d}(v) > 0 \Rightarrow r_v = 1 + \max\{r_u : (u, v) \in E_d\}$

where $G_d = (V, E_d)$. A node v with $r_v = 0$ is called a source of the valid ℓ-round target-orientation.

The proof of the following result is similar to the proof of Theorem 4.1.7 and Theorem 4.4.2.

Theorem 4.4.5 *Let $\langle G, V', t\rangle$ be an instance of the extended ℓ-round* Target Set Selection *problem. Then $S \subseteq V$ is the set of sources of a valid ℓ-round target-orientation if and only if $V' \subseteq \mathcal{P}^\ell(S)$.*

Theorem 4.4.6 *Given an instance $\langle G, V', t\rangle$ of the ℓ-round* Target Set Selection *problem where $G = (V, E)$ is an undirected graph with tree-width k, a minimum weight set $S \subseteq V$ such that $V' \subseteq \mathcal{P}^\ell(S)$ can be obtained in time $O(c^{k^2 + k\log\ell} \cdot T^{k+1} \cdot |V|)$, where c is a constant and T is the maximum threshold of nodes. Moreover, if G is a planar graph then the algorithm has the running time of $O(c^{k\log\ell} \cdot T^{k+1} \cdot |V|)$.*

Theorem 4.4.7 *Let ℓ be a given parameter, where $\ell = O(\frac{\log n}{\log\log n})$. Then there is a PTAS for the weighted ℓ-round* Target Set Selection *problem on planar graphs.*

Proof: The algorithm is the same as Algorithm 2 for the ℓ-round General Propagation problem. We only need to change the decompositions in the following way. The set $C_{i,j}$ is defined as before. Recall that $B_{i,j}$ had 2ℓ extra levels around $C_{i,j}$. Here, we only need ℓ extra levels around $B_{i,j}$. Hence, we define $B_{i,j} = L[jk+i-\ell, (j+1)k+i-1+\ell]$. The rest of algorithm is the same as before. The analysis is also similar to the analysis of the PTAS for the ℓ-round General Propagation problem. □

Part II

Packing Steiner trees

Chapter 5

Packing Steiner trees

5.1 Introduction

In the Steiner Tree Packing problem we are given a graph $G = (V, E)$ and a subset of nodes $R \subseteq V$; each node in R is called a terminal node. The terminal nodes are called *black* nodes, and the non-terminal nodes are called *white* nodes or *Steiner* nodes. An edge between two white nodes is called a *white* edge. A Steiner node or an edge is called an *element*. A tree that contains all terminal nodes in R is called an R-Steiner tree (or Steiner tree, for short). The goal in this problem is to find a set of disjoint R-Steiner trees of maximum cardinality. We study two versions of the problem: In the *edge-disjoint* version, the Steiner trees are required to be edge-disjoint; that is, each edge of the graph should be in at most one Steiner tree. In the *element-disjoint* version, the Steiner trees are required to be element-disjoint; that is, each element should be in at most one tree.

We say that two nodes are k-edge connected if there exist k edge-disjoint paths between them. Similarly, we say that two nodes are k-element connected if there exist k element-disjoint paths between them. The set of terminals, R, is called k-edge connected (or k-element connected) if each pair of terminal nodes in R is k-edge connected (or k-element connected). Without loss of generality, for both versions of the Steiner Tree Packing problem, we can assume that there are no edges between terminal nodes, by subdividing edges if needed. Note that subdividing an edge doesn't change the connectivity between terminal nodes. This operation doesn't also change the optimum value of both versions of the Steiner Tree Packing problem.

There are two well-known special cases of the Steiner tree packing problem. The first special case has $|R| = 2$, say $R = \{s, t\}$. Then the goal is to find a maximum set of edge-disjoint s-t paths (or, openly disjoint s-t paths). This problem can be solved via max-flow algorithms, and the optimal value is described by Menger's theorem [51].

Theorem 5.1.1 (Menger [51]) *Let $G = (V, E)$ be a graph and $s, t \in V$. Then* *the minimum number of edges whose deletion separates s from t is equal to the maximum number of edge-disjoint s-t paths in G.*

Another important case is when all of the nodes are terminal nodes (i.e., $R = V$); in this case we have the edge-disjoint Spanning Tree Packing problem. Tutte [71] and Nash-Williams [57] independently proved the following min-max theorem for this special case.

Theorem 5.1.2 (Tutte and Nash-Williams) *An undirected graph G has k edge-disjoint spanning trees if and only if for any partition $\mathcal{P}$ of V into $|\mathcal{P}|$ non-empty subsets we have $e(\mathcal{P}) \geq k(|\mathcal{P}| - 1)$, where $e(\mathcal{P})$ is the number of edges in G with end-nodes in different sets of $\mathcal{P}$.*

It follows easily form this theorem that a k-edge connected graph has $\lfloor \frac{k}{2} \rfloor$ edge-disjoint spanning trees. Kriesell [45] conjectured that this also extends to the edge-disjoint Steiner Tree Packing problem.

Conjecture 5.1.3 (Kriesell [45]) *If the set of terminal nodes, R, is k-edge connected in G, then there exist $\lfloor \frac{k}{2} \rfloor$ edge-disjoint R-Steiner trees in G.*

Kriesell considered a special case of the conjecture in which each Steiner node has even degree [45], and he gave a simple proof for that case. Frank, Kiraly and Kriesell [27] considered another special case and proved the following: if there are no white edges (i.e., the Steiner nodes form an independent set) and the terminals are $3k$-edge connected, then there exist k edge-disjoint Steiner trees. This is proved using a generalization of the Tutte–Nash-Williams theorem to hypergraphs [27]. Now, consider arbitrary graphs and Kriesell's (general) conjecture. Jain, Mahdian and Salavatipour [36] designed an approximation algorithm with a guarantee of $O(\frac{|R|}{4})$. Later on, Lau [46, 47] using the result of Frank et al. [27] proved that if the terminals are $24k$-edge connected, then there exist k edge-disjoint Steiner trees. This gives an approximation algorithm with a guarantee of 24.

Kaski [39] proved that the problem of finding two edge-disjoint Steiner trees is NP-hard, and also showed that the edge-disjoint Steiner Tree Packing problem is NP-hard even with 7 terminals. Jain, Mahdian and Salavatipour [36] studied an LP relaxation of the edge-disjoint Steiner Tree Packing problem and proved that the edge-disjoint Steiner Tree Packing problem is APX-hard. Cheriyan and Salavatipour [17] proved that the problem is APX-hard even with 4 terminal nodes.

Cheriyan and Salavatipour [18] studied the element-disjoint Steiner Tree Packing problem and designed an approximation algorithm with a guarantee of $O(\log n)$ for the problem. They [17] also proved that the element-disjoint Steiner Tree Packing problem is hard to approximate within a factor of $\Omega(\log n)$.

In this chapter, we study approximation algorithms and hardness results for both versions of the Steiner Tree Packing problem on planar graphs. We also study standard linear programming relaxations of both versions of the problem. The main results in this chapter are as follows:

- In Section 5.2, we prove that if we have a planar graph such that the terminal nodes are k-element connected, then there exist at least $\lfloor \frac{k}{2} \rfloor - 1$ element-disjoint Steiner trees. This provides an approximation algorithm with a guarantee of (almost) 2 for the element-disjoint version of the problem on planar graphs. Based on this, we get an approximation algorithm with a guarantee of (almost) 4 for the edge-disjoint version of the problem on planar graphs.
- In Section 5.3, we prove that both versions of the Steiner Tree Packing problem are NP-hard even with three terminals (i.e., $|R| = 3$). Moreover, based on a major recent result of Naves [58, 59], we show that the edge-disjoint version of the problem is NP-hard even in planar graphs with 3 terminals on the same face of the embedding.
- In Section 5.4, we show that even on planar graphs the standard LP relaxation of the edge-disjoint Steiner Tree Packing problem has integrality ratio at least $2-\epsilon$, where the additive term ϵ is a function of k and R and for fixed $|R|$, $\epsilon \to 0$ as $k \to \infty$. The edge-connectivity between terminal nodes is an upper bound on the objective value of the standard LP relaxation of the edge-disjoint Steiner Tree Packing problem. Hence, assuming Kriesell's conjecture, the integrality ratio is at most 2. Moreover, we modify our construction to get a similar result for the element-disjoint version of the problem.

5.2 Approximation algorithms for planar graphs

5.2.1 Element-disjoint Steiner trees

The following theorem is the main result of this section.

Theorem 5.2.1 *Let $G = (V, E)$ be an undirected planar graph, and let $R \subseteq V$ be the set of terminals, and assume that the set of terminals is k-element connected. Then there are at least $\lfloor \frac{k}{2} \rfloor - 1$ element-disjoint Steiner trees in G.*

We define the Bipartite Steiner Tree Packing problem to be a subproblem of the element-disjoint Steiner Tree Packing problem such that the graph is bipartite, all terminal nodes are in one part of the bipartition, and all Steiner nodes are in the other part. Consider a planar instance of the element-disjoint Steiner Tree Packing problem, i.e., the associated graph is planar. We can transform it into a planar instance of Bipartite Steiner Tree Packing by using the following theorem. The theorem is due to Hind and Oellermann, see [34], and a short proof is given in [18].

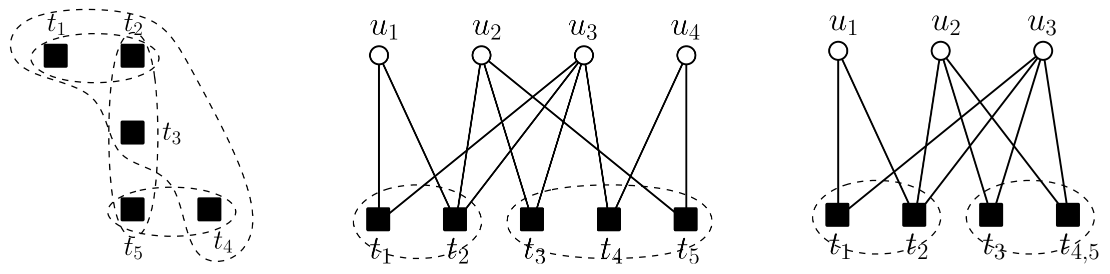
Theorem 5.2.2 *[34] Consider a graph $G = (V, E)$ that has a set of terminals R such that R is k-element connected. There is a polynomial-time algorithm that repeatedly deletes or contracts white edges to obtain a bipartite graph G' from G such that R stays k-element connected, and moreover, R forms one part of the bipartition of G'.*

A hypergraph is a pair $\mathcal{H} = (V, \mathcal{E})$ where V is the node-set of $\mathcal{H}$ and $\mathcal{E}$ is a collection of non-empty subsets of V. A subset $Z \in \mathcal{E}$ is called a *hyperedge* of $\mathcal{H}$. Given a partition $\mathcal{P} = \{V_1, \dots, V_t\}$ of V into non-empty subsets, a hyperedge $Z \in \mathcal{E}$ is called a *crossing* hyperedge if it intersects at least two subsets of $\mathcal{P}$ and otherwise it is called an *internal* hyperedge. We use $|\mathcal{P}|$ to denote the number of sets V_i in $\mathcal{P}$, and we denote the number of crossing hyperedges corresponding to the partition $\mathcal{P}$ by $e_{\mathcal{H}}(\mathcal{P})$ (or simply, by $e(\mathcal{P})$). Given a hypergraph $\mathcal{H} = (V, \mathcal{E})$, we associate a bipartite graph $G_{\mathcal{H}} = (V, U; E)$ to $\mathcal{H}$ as follows. Corresponding to each hyperedge $Z \in \mathcal{E}$ we have a node $u_Z \in U$. A node $v \in V$ is adjacent to $u_Z \in U$ if $v \in Z$; note that the degree of u_Z in $G_{\mathcal{H}}$ is the size of Z.

Consider the hypergraph $\mathcal{H}$ shown in Figure 5.1(a). The node-set of $\mathcal{H}$ is $V = \{t_1, t_2, t_3, t_4, t_5\}$, and the hyperedges of $\mathcal{H}$ are $Z_1 = \{t_1, t_2\}$, $Z_2 = \{t_2, t_3, t_5\}$, $Z_3 = \{t_1, t_2, t_3, t_4\}$ and $Z_4 = \{t_4, t_5\}$. Figure 5.1(b) shows the bipartite graph, $G = (V, U; E)$, associated with $\mathcal{H}$. Consider the partition $\mathcal{P} = \{\{t_1, t_2\}, \{t_3, t_4, t_5\}\}$ of V; this partition is shown in dashed lines in Figure 5.1(b). The hyperedges Z_2, Z_3 are crossing hyperedges w.r.t. (with respect to) $\mathcal{P}$, and hyperedges Z_1, Z_4 are internal hyperedges w.r.t. $\mathcal{P}$. Thus, $e(\mathcal{P}) = 2$, since there are two crossing hyperedges in $\mathcal{P}$. Given a partition $\mathcal{P}$, a useful operation is to *contract* an internal hyperedge: we identify all nodes in Z into a single node and remove Z from the hypergraph. For example, Figure 5.1(c) shows the bipartite representation of the hypergraph obtained by contracting the internal hyperedge $Z_4 = \{t_4, t_5\}$. If we further contract $Z_1 = \{t_1, t_2\}$ we get a copy of $K_{2,3}$. If we contract some internal hyperedges (w.r.t. $\mathcal{P}$) of $\mathcal{H}$, then we obtain a "shrunk" hypergraph $\mathcal{H}'$ and a partition $\mathcal{P}'$ of $V(\mathcal{H}')$; note that the crossing hyperedges of $\mathcal{H}$ (w.r.t. $\mathcal{P}$) are the same as the crossing hyperedges of $\mathcal{H}'$ (w.r.t. $\mathcal{P}'$).

Let $G = (R, U; E)$ be an instance of the Bipartite Steiner Tree Packing problem, where R is the set of terminal nodes and U is the set of Steiner nodes. We associate a hypergraph $\mathcal{H}_G = (R, \mathcal{E})$ to G as follows. The terminal nodes of G are the nodes in $\mathcal{H}_G$, and corresponding to each Steiner node $u \in U$ we have a hyperedge Z_u that contains the set of neighbors of u in G. Also, given any hypergraph $\mathcal{H}$, we may view its associated graph $G_{\mathcal{H}}$ as an instance of the Bipartite Steiner Tree Packing problem.

A hypergraph $\mathcal{H}$ is *k-partition connected* if $e_{\mathcal{H}}(\mathcal{P}) \geq k(|\mathcal{P}|-1)$ for every partition $\mathcal{P}$ of V. A 1-partition connected hypergraph is simply called *partition-connected.* If a hypergraph $\mathcal{H}$ is partition-connected, then it is easy to see that the associated bipartite graph $G_{\mathcal{H}}$ is connected, and so it contains a Steiner tree (with terminal set $V(\mathcal{H})$). But the converse does not hold: for a connected instance of Bipartite Steiner Tree Packing, the associated hypergraph may not be



(a) Hypergraph (b) Bipartite representation (c) Contracting $\{t_4, t_5\}$

Figure 5.1: A Hypergraph and its bipartite representation

partition-connected. Frank et al. [27] proved the following generalization of the Tutte–Nash-Williams theorem.

Theorem 5.2.3 (Theorem 2.8 in [27]) *A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is k-partition connected if and only if there is a partition $\mathcal{E}_1, \ldots, \mathcal{E}_k$ of $\mathcal{E}$ into k subsets such that each of the sub-hypergraphs $\mathcal{H}_i = (V, \mathcal{E}_i)$ is partition-connected.*

Therefore, we can obtain ℓ element-disjoint Steiner trees in G if $\mathcal{H}_G$ is ℓ-partition connected. Now we prove the following lemma that completes the proof of Theorem 5.2.1.

Lemma 5.2.4 *Let $G = (R, U; E)$ be a bipartite planar graph such that R is k-element connected. Then the hypergraph $\mathcal{H}_G = (R, \mathcal{E})$ associated with G is $\lfloor \frac{k-2}{2} \rfloor$-partition connected.*

Proof: We may assume that G is connected. Consider the hypergraph $\mathcal{H}$ and define the *fractional* partition-connectivity, λ^*, as follows:

$$\lambda^* = \min_{\mathcal{P}} \frac{e(\mathcal{P})}{|\mathcal{P}| - 1}, \tag{5.1}$$

where the minimum is over all partitions $\mathcal{P}$ of R with $|\mathcal{P}| \geq 2$. Let λ denote the partition-connectivity of $\mathcal{H}$. It follows from the definition of partition-connectivity that $\lambda = \lfloor \lambda^* \rfloor$. Let $\mathcal{P}^* = \{X_1, X_2, \ldots, X_\ell\}$ be a partition that achieves the minimum ratio λ^*. In the rest of the proof, except where mentioned otherwise, crossing hyperedges and internal hyperedges are w.r.t. $\mathcal{P}^*$.

Consider the Steiner nodes of G that correspond to the internal hyperedges. We contract all the edges of G that are incident to these Steiner nodes, and we call the resulting graph G'. In more detail, consider each internal hyperedge $Z_u \in \mathcal{E}$ and contract all edges in G adjacent to the Steiner node u corresponding to hyperedge Z_u. We may ignore all parallel edges in G' formed by these edge contractions.

Claim 5.2.5 *The obtained graph G' is a bipartite planar graph and has the following properties:*

1. *All of the remaining Steiner nodes in G' correspond to crossing hyperedges in $\mathcal{H}$, and they form one part of the bipartition,*

2. *The other part of the bipartition has $|\mathcal{P}^*|$ nodes, and each node has degree at least k.*

Proving this claim completes the lemma. This follows because G' has at least $k\,|\mathcal{P}^*|$ edges and at most $2(e(\mathcal{P}^*) + |\mathcal{P}^*|) - 4$ edges; a planar bipartite graph on n nodes has at most $2n - 4$ edges (this is a standard fact about planar bipartite graphs, see for example Diestel's book [23]). Hence, we have

$$k\,|\mathcal{P}^*| \leq 2(e(\mathcal{P}^*) + |\mathcal{P}^*|) - 4 \Longrightarrow e(\mathcal{P}^*) \geq \frac{(k-2)\,|\mathcal{P}^*|}{2} + 2 \Longrightarrow \lambda^* > \frac{k-2}{2}.$$

Proof of Claim 5.2.5: Consider a set $X_i \in \mathcal{P}^*$ of size at least 2 and arbitrarily partition it into two non-empty sets X_i' and X_i'', and let $\mathcal{P}'$ be the obtained partition. Since $\mathcal{P}^*$ is the minimum ratio partition, we have $\lambda' = \frac{e(\mathcal{P}')}{|\mathcal{P}'|-1} \geq \lambda^*$. Hence, $e(\mathcal{P}') \geq \lambda^*(|\mathcal{P}'|-1) > \lambda^*(|\mathcal{P}^*|-1) = e(\mathcal{P}^*)$. Hence, there exists a hyperedge that is crossing w.r.t. $\mathcal{P}'$ but is not crossing w.r.t. $\mathcal{P}^*$; that is, one of the internal hyperedges w.r.t. $\mathcal{P}^*$ intersects both X_i' and X_i''. This reasoning applies to each set $X_i \in \mathcal{P}^*$ and to each 2-partition X_i', X_i'' of X_i; hence, for each $X_i \in \mathcal{P}^*$, the subgraph of G induced by X_i and the Steiner nodes corresponding to the hyperedges internal to X_i is connected. Thus, contracting all edges in G adjacent to the Steiner nodes corresponding to the internal hyperedges (w.r.t. $\mathcal{P}^*$) will shrink each set X_i of $\mathcal{P}^*$ into a single node. The obtained graph G' is planar, and it is easy to see that it is bipartite with all the Steiner nodes corresponding to the crossing hyperedges (w.r.t $\mathcal{P}^*$) in one part of the partition and all of the "contracted" nodes in the other part. Now we prove that the degree of each contracted node is at least k using the fact that the terminals are k-element connected in G. To see this, consider a shrunk node v_i corresponding to a subset $X_i \in \mathcal{P}^*$, and assume that it has less than k neighbors in G'. Let Y' be the set of neighbors of v_i, so $|Y'| < k$. Note that Y' separates v_i from any other contracted node v_j in G', i.e., v_i and v_j are in different connected components of $G' \setminus Y'$. Now focus on the original hypergraph $\mathcal{H}$ and note that Y' (viewed as a subset of $\mathcal{E}(\mathcal{H})$) contains all hyperedges that intersect both X_i and $R \setminus X_i$; thus, in the original graph G, we see that Y' (viewed as a subset of U) separates X_i from the rest of the terminals, because Y' contains all Steiner nodes that are adjacent to both X_i and $R \setminus X_i$. This is a contradiction because the terminals are k-element connected in G; that is, for any set of white nodes Y whose deletion separates a pair of terminals, we must have $|Y| \geq k$. This shows that each contracted node has degree at least k in G'. □

This completes the proof. □

Now, we describe an algorithm that given a planar graph G, where the set of terminals is k-element connected, it finds at least $\left\lfloor \frac{k}{2} \right\rfloor - 1$ element-disjoint Steiner trees.

Algorithm:

1. In the first step, we reduce the given graph $G = (V, E)$ to an instance G' of the Bipartite Steiner Tree Packing problem using Theorem 5.2.2; note that G' is obtained from G by removing or contracting white edges in G.

2. In the second step, using results of Frank et al. [27], we decompose the associated hypergraph $\mathcal{H}$ of G', using Edmonds' matroid partition algorithm, into the maximum number of partition-connected sub-hypergraphs. The independence test in this algorithm is to check if the given hypergraph is a hyperforest. (See the running time analysis given below for more discussion.)

3. Each partition-connected sub-hypergraph corresponds to an Steiner tree in G'. By "uncontracting" the edges in G that were contracted in the first step of the algorithm, we obtain at least $\lfloor \frac{k}{2} \rfloor - 1$ element-disjoint Steiner trees in G.

Running time of the above algorithm: In the first step, we take a white edge e and delete it from G. Then we check if the terminals are still k-element connected. If they are k-element connected, then we take another white edge and continue, otherwise, we identify the end-nodes of e and move to the next white edge. The k-element connectivity can be tested in time $O(kn\,|R|)$ for planar graphs using the augmenting-paths algorithm for the maximum s-t flow problem. Hence, the total running time of the first step is $O(kn^2\,|R|)$, since the number of white edges is $O(n)$.

In the second step, we decompose $\mathcal{H}$ into the maximum number of disjoint partition-connected sub-hypergraphs via Theorem 5.2.3. The proof of this theorem is based on Edmonds' matroid partition theorem, and the proof can be "implemented" via the matroid partition algorithm. The running time of the matroid partition algorithm is $O(n^{2.5}f(n))$, where n denotes the size of the groundset of the matroid, and $f(n)$ denotes the running time for testing independence in the given matroid. The test for independence corresponds to testing whether a hypergraph satisfies the conditions for a hyperforest, and this in turn corresponds to testing whether a bipartite graph $(R, U; E)$ has positive surplus, i.e., $\forall S \subseteq U : |\Gamma(S)| > |S|$. It is easy to show that $f(n) = O(n^3)$ for an arbitrary bipartite graph, and for planar bipartite graphs this improves to $f(n) = O(n^2)$; this can be done using matching algorithms in bipartite graphs. Hence, the running time of our algorithm on planar graphs is $O(kn^2\,|R| + n^{4.5}) = O(n^{4.5})$.

5.2.2 Edge-disjoint Steiner trees

Theorem 5.2.6 *Let $G = (V, E)$ be an undirected planar graph, let $R \subseteq V$ be the set of terminals, and assume that R is k-edge connected. Then there are at least $\lfloor \frac{k}{4} \rfloor - 1$ edge-disjoint Steiner trees in G.*

Proof: We first reduce G to a planar graph G' with Steiner nodes of degree at most 4. This is done by repeatedly replacing a Steiner node of degree more than 4 by

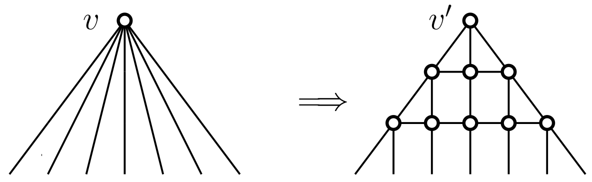


Figure 5.2: Gadget for high degree Steiner nodes

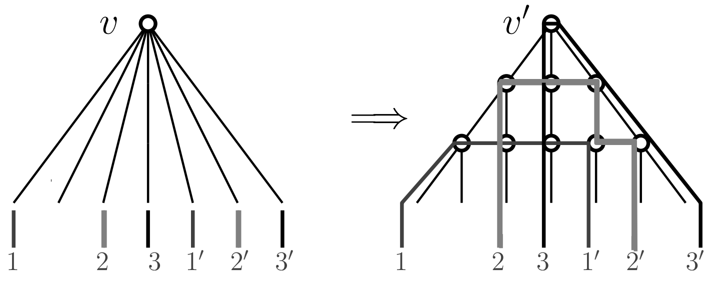


Figure 5.3: Routing the paths via the gadget

a gadget that preserves the connectivity and planarity, but doesn't introduce any new Steiner nodes of degree more than 4. Let v be a Steiner node of degree $d > 4$ in G. We replace node v by the gadget shown in Figure 5.2. The gadget has $\lceil \frac{d}{2} \rceil - 1$ rows including the last row containing a single node v'. Clearly, the obtained graph has one less Steiner node of degree more than 4; also, the set of terminal nodes and their degrees stay the same. Moreover, R is k edge-connected in the obtained graph. To show this, we claim that any set of paths using edges incident to v can be rerouted via the gadget. We sketch a proof of this claim, although the gadget and its properties are well known, see [53, 59]. This can be proved by induction on the number of rows in the gadget. The base case is when there is only one row containing the single node v', and note that v' has degree at most 4. Clearly for this case the claim is true. Given a paring of edges used in the paths going through v, we first route one of the extreme pairs (i.e., the pair using the left most edge or the pair using the right most edge), say the left most pair, using the first horizontal row of the gadget. Next, we send the other pairs using the vertical edges to the next horizontal row. Some of the paths may need to be "shifted" to the left; for example, the path labeled $(2, 2')$ is shifted on the first row to the left in Figure 5.3. Finally, we inductively reroute the remaining paths.

We replace all Steiner nodes of degree more than 4 to get a planar graph G' with no Steiner node of degree more than 4 such that the terminal nodes are k-edge connected in G'. Also observe that edge-disjoint Steiner trees in G' can be transformed to edge-disjoint Steiner trees in G.

We can assume that G' has no edge connecting two terminals; such an edge can be subdivided by introducing a Steiner node. Now we show that the terminal nodes in G' are $\lceil \frac{k}{2} \rceil$ element-connected. Let X be a set of white nodes of minimum cardinality whose deletion separates some two terminals, i.e., $G'\backslash X$ has at least two connected components $C_1, C_2, \ldots, C_\ell$ that each contains a terminal. There are at most $4|X|$ edges going out of X since each white node in X has degree ≤ 4. Hence, there is a component C_j that has at most $2|X|$ edges entering it, and consequently there is an edge-cut of size at most $2|X|$ that separates a pair of terminals. Since such a cut has size $\geq k$, we have $|X| \geq \frac{k}{2}$. This shows that terminal nodes are $\lceil \frac{k}{2} \rceil$ element-connected in G'. Now we apply Theorem 5.2.1 to G' and obtain $\lfloor \frac{k}{4} \rfloor - 1$ element-disjoint Steiner trees in G'. These Steiner trees are clearly edge-disjoint. Thus, G has at least $\lfloor \frac{k}{4} \rfloor - 1$ edge-disjoint Steiner trees. □

5.3 Hardness results

5.3.1 General graphs

In this subsection, we prove the following hardness result.

Theorem 5.3.1 *The element-disjoint* STEINER TREE PACKING *problem with* 3 *terminals is NP-hard.*

We prove our result by a reduction from the 2DIRPATH problem. In the 2DIRPATH problem, we are given a directed graph $G = (V, E)$ with two pairs (x_1, y_1), (x_2, y_2) of nodes. The goal in this problem is to find two node-disjoint paths P_1 and P_2 such that P_i, for $i = 1, 2$, is a directed path in G from x_i to y_i. We denote this instance of the problem by $\mathcal{I} = (G, x_1, y_1, x_2, y_2)$.

Theorem 5.3.2 ([26]) *It is NP-complete to decide if a given instance of the* 2DIRPATH *problem has a solution.*

Reduction: Let $\mathcal{I} = (G, x_1, y_1, x_2, y_2)$ be an instance of the 2DIRPATH problem. We construct an undirected graph G' (with three terminal nodes r, t_1, t_2) from G as follows. All the nodes of G are kept in G', and we denote these nodes by the same label in both graphs.

1. Add the following new nodes r, t_1, t_2, x'_1, x'_2 to G. The terminal nodes in G are $R = \{r, t_1, t_2\}$.

2. In the next step we install copies of the gadget given in Figure 5.4. The unlabeled nodes and the nodes labeled by u', v' are called *internal* nodes of the gadget, and the rest of nodes are called *external* nodes. The internal nodes of each gadget are disjoint from the internal nodes of the other gadgets.

3. Replace each directed edge $(u, v) \in E(G)$ by a new copy of the gadget shown in Figure 5.4.

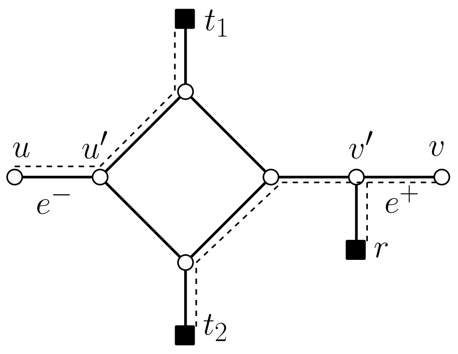


Figure 5.4: Gadget corresponding to a directed edge $e = (u, v) \in E(G)$

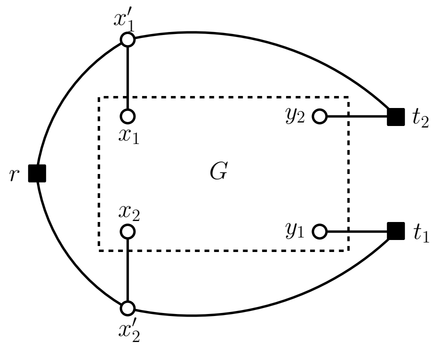


Figure 5.5: The constructed graph G' with 3 terminals

4. Add the following (undirected) edges to G: $\{x'_1, x_1\}$, $\{x'_2, x_2\}$, $\{r, x'_1\}$, $\{r, x'_2\}$, $\{x'_1, t_2\}$, $\{x'_2, t_1\}$, $\{y_1, t_1\}$, $\{y_2, t_2\}$. Let G' be the obtained undirected graph. (See figure 5.5 for an illustration).

The following lemma completes the proof of Theorem 5.3.2.

Lemma 5.3.3 *The instance $\mathcal{I}$ of the* 2DirPath *problem has a solution if and only if there are $\lambda = |E(G)| + 2$ element-disjoint Steiner trees in G'.*

Proof: Observe that using the internal nodes of each gadget we can find a Steiner tree. We call such a tree a *local* Steiner tree. Hence, G' has at least $|E(G)|$ element-disjoint Steiner trees. Also note that each terminal has degree $\lambda = |E(G)| + 2$, so there are at most λ element-disjoint Steiner trees in G'.

Assume that $\mathcal{I}$ is a yes instance and P_1, P_2 are two node-disjoint directed paths in G from x_1 to y_1, and x_2 to y_2 respectively. We show that G' has λ element-disjoint Steiner trees. First, make $|E(G)|$ element-disjoint local Steiner trees corresponding to the directed edges in G. Next, we form two partial Steiner trees $\mathcal{T}_1 = \{\{r, x'_1\}, \{x'_1, t_2\}, \{x'_1, x_1\}\}$ and similarly $\mathcal{T}_2 = \{\{r, x'_2\}, \{x'_2, t_1\}, \{x'_2, x_2\}\}$. Note that $\mathcal{T}_i$ consists of the 3 edges incident to x'_i, and we need to add a path from

x_i to t_i. Now we augment the partial Steiner trees $\mathcal{T}_1, \mathcal{T}_2$ by modifying the local Steiner trees of the directed edges on P_1, P_2. For each edge $e = (u, v) \in P_1$ we form two partial Steiner trees in the gadget corresponding to e. The first one is a path of length three from u to t_1, and the second tree is obtained from the path of length four from t_2 to v by adding the edge $\{r, v'\}$ to it; these two partial Steiner trees are denoted by dashed lines in Figure 5.4. Now, we combine partial Steiner trees as follows. First, combine the partial Steiner tree $\mathcal{T}_1$ with the first partial Steiner tree corresponding to the first directed edge of P_1. Next, combine the second partial Steiner tree corresponding to each edge of P_1 with the first partial Steiner tree corresponding to the next edge of P_1. Finally, add the edge $\{y_1, t_1\}$ to the second partial tree corresponding to the last edge of P_1. This completes $\mathcal{T}_1$ as well as all partial trees corresponding to the edges of P_1. Similarly, we can complete the partial Steiner tree $\mathcal{T}_2$ and all the partial Steiner trees corresponding to the edges of P_2. Clearly, any two of these Steiner trees are element-disjoint by our construction and by the fact that P_1, P_2 are node-disjoint. Therefore, G' has λ element-disjoint Steiner trees.

Now assume that G' has λ element-disjoint Steiner trees. We prove that G has two node-disjoint directed paths P_1 and P_2 where each P_i is an x_i-y_i path. Any two edges incident to a terminal must be in two different Steiner trees since each terminal has degree λ in G'. Thus, we may assume that each Steiner tree has the three terminals as leaf nodes and has no other leaf nodes; that is, each Steiner tree is a subdivision of $K_{1,3}$. Therefore, x'_1 and x'_2 each has degree three in two different Steiner trees. Let us denote the Steiner tree that contains x'_i by $\mathcal{T}_i$. Observe that $\mathcal{T}_1$ has an x'_1-t_1 path via x_1, and similarly $\mathcal{T}_2$ has an x'_2-t_2 path via x_2. In order to find P_1 and P_2 we label the edges in G' that are incident to the original nodes of G; for example, in Figure 5.4 e^- and e^+ are two such edges. Consider the gadget corresponding to $e = (u, v) \in E(G)$. If the edge e^- adjacent to u in this gadget is not used in any Steiner tree, then label e^- by $L(e^-) = \emptyset$. Otherwise, consider the Steiner tree, $\mathcal{T}$, containing e^-. Let $S \subseteq \{r, t_1, t_2\}$ be the set of terminals in the component of $\mathcal{T} - e^-$ containing u. Assign the label $L(e^-) = S$ to e^-. Similarly, we can assign a label $L(e^+)$ to the edge e^+ (the edge incident to v) by considering the component in $\mathcal{T}' - e^+$ containing v', where $\mathcal{T}'$ is the Steiner tree containing e^+.

In the rest of the proof, we assume w.l.o.g. (without loss of generality) that no Steiner tree uses both edges e^-, e^+ of a gadget. Suppose there is a Steiner tree $\mathcal{T}$ that uses both edges e^-, e^+ of a gadget. Then no other Steiner tree can "enter" this gadget. Thus, we can replace $\mathcal{T}$ by a local Steiner tree of this gadget. This preserves the number of element-disjoint Steiner trees.

Claim 5.3.4 *In the gadget corresponding to an edge $(u, v) \in E(G)$ we have*

$$L(e^-) = L(e^+) \in \{\emptyset, \{r\}, \{r, t_1\}, \{r, t_2\}\}.$$

Proof: Observe that each terminal $t \in \{r, t_1, t_2\}$ is incident to exactly one edge of the gadget, and denote that edge by $e(t)$. Moreover, each of these three edges must be in one of the Steiner trees. First, note that if either $L(e^-) = \emptyset$ or $L(e^+) = \emptyset$, then both labels are the same. Otherwise, at least one of the three edges incident to the terminals in the gadget cannot be in any Steiner tree. Now assume that $L(e^-) \neq L(e^+)$, and let the two Steiner trees using e^- and e^+ be $\mathcal{T}^-, \mathcal{T}^+$. Now focus on the components in $\mathcal{T}^- - e^-$ and $\mathcal{T}^+ - e^+$ containing u and v, respectively, and observe that the terminal sets in these two components are $L(e^-)$ and $R \setminus L(e^+)$. Since $L(e^-) \neq L(e^+)$ there is a terminal t that either is missing from these two components or appears in both of them. This is not possible because in the first case, the edge $e(t)$ cannot be present in both Steiner trees, and in the second case, $e(t)$ cannot be used in any other Steiner tree. Thus, $L(e^-) = L(e^+)$. Observe that the Steiner tree using edge e^+ must contain the edge $e(r) = \{r, v'\}$ otherwise this edge cannot be used in any Steiner tree. Hence, the label assigned to e^+ is either $\emptyset$ or it contains the terminal r. This completes the proof. $\square$

This labeling of edges in G' induces a labeling for the directed edges in G; label the directed edge $(u, v) \in E(G)$ by the label of e^-, $L(e^-)$, in the gadget corresponding to (u, v). By the above claim, this is the same as the label of e^+ in that gadget.

Claim 5.3.5 *Let $e = (u, v)$ be a directed edge in G such that $v \notin \{y_1, y_2\}$ and the label of e is in $\{\{r, t_1\}, \{r, t_2\}\}$. Then there is an out-going edge from v, call it f, that has the same label as e, and all other edges incident to v (in-coming or out-going) have the label $\emptyset$.*

Proof: Let $e = (u, v)$ be an edge with label $\Lambda = \{r, t_1\}$; the proof for the other case is similar. Consider the gadget corresponding to e in G', and let $\mathcal{T}$ be the Steiner tree containing the edge e^+. Note that $\mathcal{T}$ is connected to r, t_1 in this gadget, so it must be connected to t_2 via node v. Hence, v has degree 2 in $\mathcal{T}$. Let f be the directed edge incident to v whose gadget contains the remaining edges of $\mathcal{T}$. The edge f is in G since $v \notin \{y_1, y_2\}$. If f is an in-coming edge to v, then it is labeled by $\{t_2\}$ but this is not possible by Claim 5.3.4. Hence, f is an out-going edge of v, and it follows from the definition of the labels that f has the same label as e. $\square$

Now, we claim that there is a directed path in G from x_1 to y_1 whose edges are labeled by $\{r, t_2\}$, and similarly, there is a directed path in G from x_2 to y_2 whose edges are labeled by $\{r, t_1\}$. Consider the tree $\mathcal{T}_1$ containing x_1'; it must use one of the directed edges of G incident to x_1, so either there is an out-going edge from x_1 labeled $\{r, t_2\}$ or an in-coming edge to x_1 labeled $\{t_1\}$; but the latter is ruled out by Claim 5.3.4. Similarly, one of the directed edges of G going out of x_2 is labeled $\{r, t_1\}$. Hence, by Claim 5.3.5, there is a directed path P_1 starting from x_1 whose edges are labeled by $\{r, t_2\}$, and that can only end at either y_1 or y_2; but the latter is not possible, otherwise the edge $\{y_2, t_2\}$ cannot be in any Steiner tree. Thus, P_1 is a directed path from x_1 to y_1. Similarly, there is a directed path P_2 from x_2 to y_2. It also follows from Claim 5.3.5 that P_1 and P_2 are node-disjoint. $\square$

5.3.2 Planar graphs

In this section, we prove the following result.

Theorem 5.3.6 *The edge-disjoint* STEINER TREE PACKING *problem in planar graphs with* 3 *terminals is* NP*-hard even with all terminals on the outer-face.*

We reduce the EDGE-DISJOINT PATH problem with 2 demand edges to the STEINER TREE PACKING problem in planar graphs with 3 terminals. Let $\mathcal{I} = (G; x_1, y_1, d_1; x_2, y_2, d_2)$ be an instance of the planar EDGE-DISJOINT PATH problem, where $G = (V, E)$ is a planar graph and $(x_1, y_1), (x_2, y_2)$ are demand edges. Moreover, the demand edges are in the outer-face and they are crossing; that is, adding them to G makes the obtained graph non-planar. The goal in this problem is to find d_1 paths from x_1 to y_1 and d_2 paths from x_2 to y_2 such that all of the $d_1 + d_2$ paths are edge-disjoint. Our result is based on the following major recent result of Naves [58, 59].

Theorem 5.3.7 (Theorem 9 in [58]) *The planar* EDGE-DISJOINT PATH *problem with two crossing demand pairs on the outer-face is* NP*-hard.*

Reduction:

1. Start from a copy of G and add 3 terminal nodes $\{t, t_1, t_2\}$ and two non-terminal nodes s_1, s_2 to the outer-face of G.
2. Add d_1 parallel edges from s_1 to each of t, t_2, x_1, and similarly we add d_2 parallel edges from s_2 to each of t, t_1, x_2.
3. Finally, we add d_1 parallel edges from y_1 to t_1, and d_2 parallel edges from y_2 to t_2. Note that the obtained graph is planar.
4. Let H be the obtained graph, and let $R = \{t, t_1, t_2\}$ (see Figure 5.6 for an illustration).

The following lemma completes the proof of Theorem 5.3.6.

Lemma 5.3.8 *The planar graph H has $d_1 + d_2$ edge-disjoint Steiner trees if and only if the* EDGE-DISJOINT PATH *problem in G has a solution.*

Proof: Assume that the EDGE-DISJOINT PATH problem in G has a solution; that is, there are d_1 edge-disjoint x_1-y_1 paths, and there are d_2 edge-disjoint x_2-y_2 paths, and all these $d_1 + d_2$ paths are edge-disjoint. Observe that by adding edges $\{s_1, t\}$, $\{s_1, x_1\}$, $\{s_1, t_2\}$, $\{y_1, t_1\}$ to any x_1-y_1 path in G we get a Steiner tree. Hence, using d_1 edge-disjoint x_1-y_1 paths, we get d_1 edge-disjoint Steiner trees. Similarly, we can get d_2 edge-disjoint Steiner trees using d_2 edge-disjoint x_2-y_2 paths. These

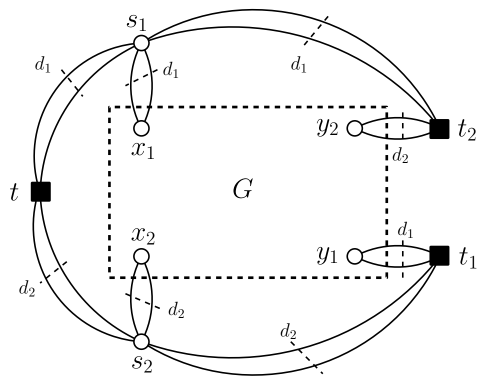


Figure 5.6: Planar graph with 3 terminals on the outer-face

$d_1 + d_2$ Steiner trees are all edge-disjoint. Thus, graph H has $d_1 + d_2$ edge-disjoint R-Steiner trees.

Now we prove the other direction. Suppose that H has $d_1 + d_2$ edge-disjoint Steiner trees. The degree of each terminal in any one of these $d_1 + d_2$ Steiner trees is one since each terminal has degree $d_1 + d_2$ in H. Therefore, each Steiner tree has a Steiner node of degree three. We claim that this degree three node is either s_1 or s_2, and moreover only one of s_1 or s_2 can be in any Steiner tree. Focus on the edges with exactly one end-node in $V(G)$; these are the edges crossing the dashed box in Figure 5.6. First we show that each Steiner tree, in the optimal packing, has exactly two of these crossing edges. To see this, consider a Steiner tree T. If T has no crossing edges, then terminal t is forced to have degree two in T, which is not possible. Also note that T cannot have only one crossing edge since there are no terminal node in $V(G)$. This shows that T has at least two crossing edges. This implies that each Steiner tree has exactly two crossing edges since there are $(d_1 + d_2)$ Steiner trees and $2(d_1 + d_2)$ crossing edges in total. Hence, the degree three node in each Steiner tree cannot be in $V(G)$ (i.e., it cannot be inside the dashed box in Figure 5.6), so the degree three node is either s_1 or s_2. Also since there are $3(d_1 + d_2)$ edges adjacent to s_1 and s_2 no tree contains both s_1 and s_2. Now consider a Steiner tree T containing s_1. Tree T uses the edge $\{s_1, x_1\}$ to enter G and it has no edges adjacent to s_2, so it must contain $\{y_1, t_1\}$. Hence, T contains a x_1-y_1 path inside G. Similarly, if T contains s_2, then it has a x_2-y_2 path inside G. Therefore, the $d_1 + d_2$ Steiner trees give us a solution to the instance $\mathcal{I}$ of the planar EDGE-DISJOINT PATH problem. □

5.4 Integrality ratio for packing Steiner trees in planar graphs

In this section, we first show that even on planar graphs the standard LP relaxation of the edge-disjoint STEINER TREE PACKING problem has integrality ratio approaching 2. This shows that any approximation algorithm, for the edge-disjoint Steiner Tree Packing problem, that uses the optimal value of this LP relaxation as an upper bound has approximation guarantee of at least 2 even on planar graphs. Moreover, most of the upper bounds known to us are no better than the optimal value of the LP relaxation. Finally, we modify our integrality ratio example to get a similar result for the element-disjoint version of the problem.

Integrality ratio example for packing edge-disjoint Steiner trees:

Consider the following linear programming relaxation of the edge-disjoint version of the problem. Let $\mathcal{T}$ be the set of all R-Steiner trees in $G = (V, E)$.

$$\begin{aligned} &(\texttt{LP-edge}) \quad z_{LP}(G) = \max \sum_{T \in \mathcal{T}} x_T \\ &\text{subject to} \\ &\qquad \sum_{T \in \mathcal{T}: e \in T} x_T \leq 1 \qquad \forall e \in E \\ &\qquad x_T \geq 0 \qquad \forall T \in \mathcal{T} \end{aligned}$$

Start from a $k \times dk$ grid and add d terminal nodes, $R = \{t_1, \ldots, t_d\}$, to the outer face of the grid. Connect terminal t_1 to the first k consecutive nodes on the bottom-most row of the grid, next, connect terminal t_2 to the second k consecutive nodes on the bottom-most row, and continue in this way to connect each terminal to a set of k consecutive nodes. Note that the set of neighbors of terminals are disjoint. Now, replace each Steiner node of degree 4 in the obtained graph by the gadget shown in Figure 5.7. Let G be the obtained graph. Note that G is a planar

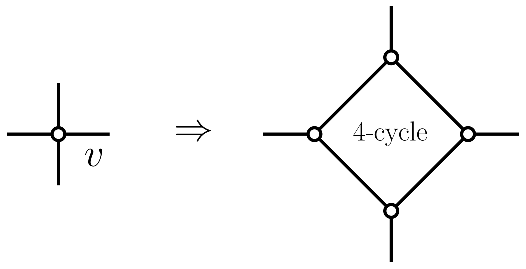


Figure 5.7: Gadget for degree 4 nodes

graph, and each Steiner node has degree at most 3. There are k edge-disjoint paths between any two terminal nodes; each path is formed by using one row and two columns of the grid. Hence, the terminal set R is k-edge-connected.

First, we prove that G has at most $\frac{kd}{2(d-1)} + \binom{d}{3}$ element-disjoint Steiner trees. Note that edge-disjoint Steiner trees in G are also element-disjoint since each Steiner node has degree at most 3 in G. Next, we show that `LP-edge` has optimal value of $z_{LP}(G) = k$. Proving these two claims shows that `LP-edge` has integrality ratio of $2 - \epsilon$.

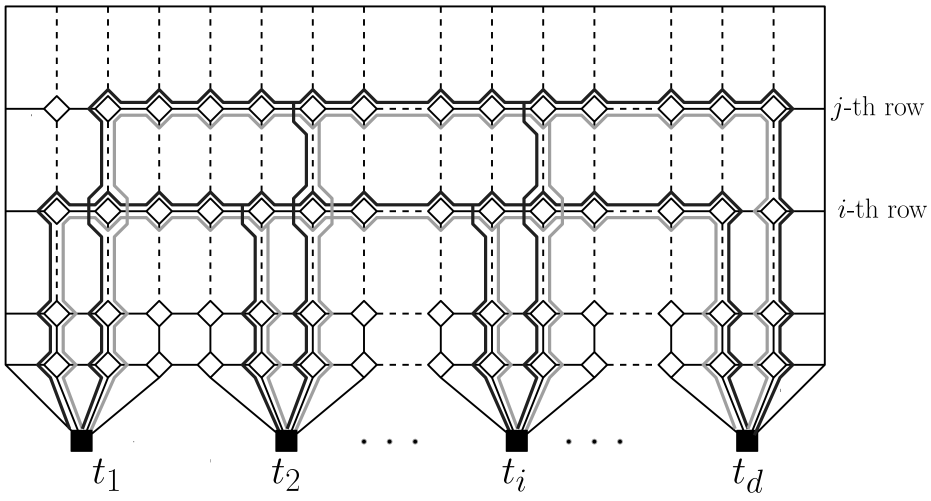


Figure 5.8: Integrality ratio example for the edge-disjoint problem

Theorem 5.4.1 *The above linear programming relaxation of the edge-disjoint* Steiner Tree Packing *problem has an integrality ratio of* $2 - \frac{2}{|R|} - \epsilon$ *even on planar graphs, where the additive term* ϵ *is a function of* k *and* $|R|$ *and for fixed* $|R|$, $\epsilon \to 0$ *as* $k \to \infty$.

Now we prove the above two claims.

Lemma 5.4.2 *Graph* G *has at most* $\frac{kd}{2(d-1)} + \binom{d}{3}$ *element-disjoint Steiner trees.*

Proof: Let $\mathcal{S}$ be a maximum-size set of element-disjoint R-Steiner trees. The edges incident to terminals are called *terminal edges.* Let $\mathcal{S}_1$ be the set of Steiner trees from $\mathcal{S}$ with less than $2(d-1)$ terminal edges, and $\mathcal{S}_2$ be the remaining subset of $\mathcal{S}$ (i.e., $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$). Note that $|\mathcal{S}_2| \le \frac{kd}{2(d-1)}$ since any Steiner tree in $\mathcal{S}_2$ has at least $2(d-1)$ terminal edges and there are kd terminal edges in total. Now, we show that $\mathcal{S}_1 \le \binom{d}{3}$. Note that each Steiner tree in $\mathcal{S}_1$ has at least 3 terminals of degree one. Hence, any Steiner tree $T \in \mathcal{S}_1$ has at least one Steiner node v of degree at least 3. Let T be a Steiner tree in $\mathcal{S}_1$ and v be a Steiner node of degree at least 3 in T. Consider the maximal subtree F of T containing v such that all leaves of F are terminals and all internal nodes of F are non-terminals; such a subtree is called a *full component* of T. Denote the leaves of the full component F containing v by $L(T, v)$, and note that $|L(T, v)| \ge 3$. Let $\mathcal{F} = \{L(T, v) | T \in \mathcal{S}_1, v \text{ is a non-terminal node}, d_T(v) \ge 3\}$. The set $\mathcal{F}$ contains all full components corresponding to non-terminal nodes of degree at least 3 in Steiner trees from $\mathcal{S}_1$. We claim that $|\mathcal{F}| \le \binom{d}{3}$; this completes the proof since $|\mathcal{S}_1| \le |\mathcal{F}|$. Note that the full components in $\mathcal{F}$ are element-disjoint. Given a set of 3 terminals $R' \subseteq R$, we show that R' can appear on the leaves of at most one subtree in $\mathcal{F}$. Note that G is a planar graph with all terminals on the outer face. Assume that there are two subgraph $F', F'' \in \mathcal{F}$ where both contain R'. Consider the union of

F', F'', and add a node on the outer face of G and connect it to R'. The obtained graph contains a subdivision of $K_{3,3}$ as a subgraph. This is a contradiction since G is a planar graph. Hence, corresponding to each subset of size three from R we have at most one subtree in $\mathcal{F}$. Therefore, $|\mathcal{F}| \leq \binom{d}{3}$. □

Lemma 5.4.3 *The linear program* `LP-edge` *has objective value $z_{LP}(G) = k$.*

Proof: The optimal value $z_{LP}(G)$ is at most k since each terminal has degree k. Now, we create $2k$ half-integral Steiner trees, and this shows that $z_{LP}(G) \geq k$. Corresponding to each row of the original grid we construct a pair of half-integral Steiner trees. The ℓth pair uses the ℓth row and columns $\ell, k+\ell, \ldots, ik+\ell, \ldots, dk+\ell$ of the grid. One of the trees in this pair uses the upper two edges of the 4-cycles and the other tree uses the lower two edges of the 4-cycles of the ℓth row. Similarly one of them uses the right two edges of the 4-cycles and the other one uses the left two edges of the 4-cycles of the columns. In Figure 5.8 Steiner trees for the ith row and the jth row are shown. It is easy to check that each edge of the modified grid is contained in at most two Steiner trees; Figure 5.8 shows how two Steiner trees cross each other. Hence, the k pairs of half-integral Steiner trees form a solution to `LP-edge` of value k. □

Integrality ratio example for packing element-disjoint Steiner trees:

Consider the following linear programming relaxation of the element-disjoint Steiner Tree Packing problem. For notational convenience, we assume there are no edges between terminals, by subdividing edges if needed. We show that `LP-element` also has an integrality ratio of $2 - \epsilon$ in planar graphs.

$$\texttt{(LP-element)} \quad z_{LP}(G) = \max \sum_{T \in \mathcal{T}} x_T$$

subject to

$$\sum_{T \in \mathcal{T}: v \in T} x_T \leq 1 \qquad \forall v \in V \setminus R$$

$$x_T \geq 0 \qquad \forall T \in \mathcal{T}$$

Start from a $2k \times 2kd$ grid and subdivide the alternate edges of the last row of the grid. Now add d terminal nodes $R = \{t_1, \ldots, t_d\}$ to the outer face of the grid. Next connect each terminal node t_i to k consecutive subdivided nodes (see Figure 5.9 for an illustration). Let G be the obtained graph. The same analysis as in Theorem 5.4.2 shows that G has at most $\frac{kd}{2(d-1)} + \binom{d}{3}$ element-disjoint Steiner trees. We claim that `LP-element` has optimal value of $z_{LP} = k$. This proof is also similar to Lemma 5.4.3. We construct k pairs of half-integral Steiner trees. Each pair is obtained from two consecutive rows by connecting each terminal to these two rows using two consecutive columns. Figure 5.9 shows two pairs of half-integral Steiner trees and shows how they cross each other. It is easy to check that each Steiner node is contained in at most 2 half-integral Steiner trees. Hence, these $2k$ Steiner

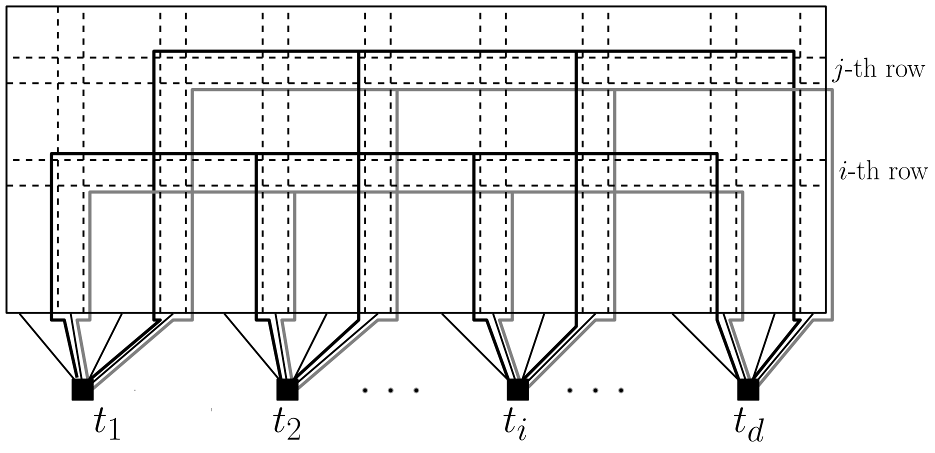


Figure 5.9: Integrality ratio example for the element-disjoint problem

trees form a feasible solution to `LP-element`. Thus, we have $z_{LP} \geq k$; moreover, $z_{LP} \leq k$ since each terminal has degree k. Therefore, the above planar example proves the following theorem.

Theorem 5.4.4 *The LP relaxation of the element-disjoint* Steiner Tree Packing *problem has an integrality ratio of* $2 - \frac{2}{|R|} - \epsilon$ *even on planar graphs, where the additive term* ϵ *is a function of* k *and* $|R|$ *and for fixed* $|R|$, $\epsilon \to 0$ *as* $k \to \infty$.

Appendices

Appendix A

Tree-width

In this section we provide standard definitions and basic properties of *tree decompositions* and related parameters.

Definition A.1 [23] *A* tree decomposition *of a graph* $G = (V, E)$ *is a pair* $\langle \{X_i \subseteq V | i \in I\}, T = (I, F)\rangle$ *such that* T *is a tree with* $V(T) = I$, $E(T) = F$, *and satisfying the following properties:*

(T1) $\bigcup_{i \in I} X_i = V$, *and every edge* $uv \in E$ *has both ends in some* X_i,

(T2) *For all* $i, j, k \in I$ *if* j *is on the unique path from* i *to* k *in* T *then we have* $X_i \cap X_k \subseteq X_j$,

The width *of* $\langle \{X_i | i \in I\}, T\rangle$ *is* $max_{i \in I} |X_i| - 1$. *The* tree-width *of* G, *denoted by* $\mathtt{tw}(G)$, *is defined as the minimum width over all tree decompositions. The nodes of* T *are called* T-nodes *and* X_i*'s are called* bags.

Definition A.2 *A* path decomposition *is a tree decomposition* $\langle \{X_i \subseteq V | i \in I\}, T = (I, F)\rangle$, *where* T *is a simple path. Similarly, the* path-width *of the graph* G, *denoted by* $\mathtt{pw}(G)$, *is defined as the minimum width over all path decompositions.*

For designing a dynamic programming algorithm based on tree decomposition, it is easier to work with a *nice* tree decomposition.

Definition A.3 *A tree decomposition* $\langle \{X_i \subseteq V | i \in I\}, T = (I, F)\rangle$, *where* T *is a rooted tree, is called a* nice *tree decomposition if the following conditions are satisfied:*

1. *every node of* T *has at most two children,*

2. *if a node* i *has two children* j *and* k, *then* $X_i = X_j = X_k$. *The node* i *is called a* Join *node.*

3. if a node i has one child j, then either

- *$X_j \subset X_i$ and $|X_i \setminus X_j| = 1$; i is called an* Introduce *node, or*
- *$X_i \subset X_j$ and $|X_j \setminus X_i| = 1$; i is called a* Forget *node.*

Let G be a given graph of tree-width k (i.e., $\mathtt{tw}(G) = k$). It is known that any tree decomposition of width k can be transformed to a nice tree decomposition with width k in linear time. Thus, we can always assume that we are given a nice tree-decomposition of the input graph.

Lemma A.4 (Lemma 13.1.3 in [42]) *Given a tree-decomposition of a graph G of width k and $O(n)$ nodes, where n is the number of nodes in G, one can find a nice tree-decomposition of G of width k and with at most $4n$ nodes in $O(n)$ time.*

Bodlaender et al. [12] showed that the tree-with and path-width can be approximated within poly-logarithmic ratio.

Theorem A.5 ([12]) *Given a graph a graph $G = (V, E)$, there exists a polynomial time algorithm that finds a tree decomposition of width at most $O(\mathtt{tw}(G) \log n)$, where $n = |V|$. Also there exists a polynomial time algorithm that finds a path decomposition of width at most $O(\mathtt{pw}(G) \log^2 n)$.*

Seymour and Thomas [69] gave a polynomial time algorithm for computing *branch-width* of planar graphs; this implies a polynomial time approximation algorithm with guarantee of $\frac{3}{2}$ for computing tree-width of planar graphs.

Appendix B

Dynamic programming algorithms

In this section we present our dynamic programming algorithms for three related problems, namely, the General Propagation problem, the extended ℓ-round General Propagation problem, and the Directed PDS problem. These algorithms are based on the dynamic programming algorithm designed by Guo et al. [30] to optimally solve PDS for undirected graphs with bounded tree-width. We also present dynamic programming algorithms for the Target Set Selection and ℓ-round Target Selection Set problems; these algorithms are very similar to the algorithms for the General Propagation problem and its extensions.

By Lemma A.4, we can assume that we are given a nice tree decomposition of the undirected graph G, $\langle \{X_i | i \in I\}, T \rangle$. Let T_i denote the subtree of T rooted at T-node i, and $Y_i = \left(\bigcup_{j \in V(T_i)} X_j \right) \setminus X_i$. We denote the subgraph of G induced on node set $Y_i \cup X_i$ by G_i (i.e., $G_i = G[Y_i \cup X_i]$).

B.1 The general propagation problem

We start by describing the state of bags in our dynamic programming algorithm.

The state of a bag: Given a valid orientation $\widehat{\mathcal{O}}$ of G, the state of a bag X_i describes the orientation of the edges in $G[X_i]$. In order to detect the dependency cycles in $\widehat{\mathcal{O}}$ without reconstructing the whole orientation, we need to store more information. This extra information enables us to detect a dependency cycle in G_i that goes through X_i, by considering only the state of the bag X_i. A *bag state* s contains the following information: 1) state of each edge, 2) state of each node, and 3) state of each pair of nodes in $G[X_i]$.

- `State of an edge`: The state of an edge $e = \{u, v\} \in E(G[X_i])$ denoted by $s(e)$ shows the orientation that is assigned to e, and has the following 3 values:

  1. $s(e) = (u, v)$: the edge e is oriented from u to v,

2. $s(e) = (v, u)$: the edge e is oriented from v to u, and
3. $s(e) = \perp$: the edge e is unoriented.

- `State of a node`: For each node $v \in X_i$ we define two states denoted by $s^-(v)$ and $s^+(v)$ that is the number of in-coming and out-going edges between v and Y_i, respectively. The state $s^-(v)$ can take any value from $\{0, 1\}$, and the state $s^+(v)$ can take any value from $\{0, 1, 2\}$. The $s^+(v) = 2$ denotes that there are at least 2 oriented edges from v to Y_i. To distinguish between propagation nodes and domination nodes we only need to know if the out-degree of a node is ≥ 2 or not. By the degree constraints in Definition 4.1.3, we cannot have a node v with in-degree 1 and out-degree > 1. Hence, these two node states have the following 5 combinations:

  1. $s^-(v) = 0, s^+(v) = 0$: There are no oriented edges between v and Y_i,
  2. $s^-(v) = 0, s^+(v) = 1$: There is no in-coming edge from Y_i, and exactly one out-going edge from v to Y_i,
  3. $s^-(v) = 0, s^+(v) = 2$: There is no in-coming edge from Y_i, and there are at least two out-going edges from v to Y_i,
  4. $s^-(v) = 1, s^+(v) = 0$: There is exactly one in-coming edge from Y_i, and no out-going edge from v to Y_i, and
  5. $s^-(v) = 1, s^+(v) = 1$: There is exactly one in-coming edge from Y_i, and one out-going edge from v to Y_i.

- `State of a pair of nodes`: We categorize dependency paths according to the type of their first and last edges. There are 4 possible types UU, UD, DU, and DD. The type xy ($x, y \in \{U, D\}$) denotes that the first edge is of type x and the last edge is of type y, where U means the edge is unoriented and D means that the edge is oriented. For example, a dependency path of type UD is a dependency path where the first edge is unoriented and the last edge is oriented. We define an operation denoted by $\otimes$ that takes the type of two dependency paths and returns the type of the dependency path that is obtained by concatenating the given two paths. Let P_1 be a dependency path of type t_1 from u to v, and P_2 be a dependency path of type t_2 from v to w. We denote by $t_1 \otimes t_2$ the type of the dependency path from u to w that is obtained by concatenating P_1 and P_2. If the concatenation of the two paths is not a dependency path, then the operator $\otimes$ returns $\emptyset$. For example, $DD \otimes UU = DU$ and $DU \otimes UD = \emptyset$; in the second example, note that after concatenating the given two paths we get two consecutive unoriented edges so the resulting path is not a dependency path. A single oriented edge (u, v) is considered to be a dependency path of type DD, and an unoriented edge $\{u, v\}$ is considered to be a dependency path of type UU.

  For a pair $(u, v) \in X_i \times X_i$ ($u \neq v$) the state of (u, v) denoted by $s(u, v)$ shows the type of dependency paths from u to v in $G[Y_i \cup \{u, v\}]$; so we have $s(u, v) \subseteq \{UU, UD, DU, DD\}$. Note that there are $2^4 = 16$ different states for each pair of nodes; $s(u, v) = \emptyset$ means that there is no dependency path from u to v in $G[Y_i \cup \{u, v\}]$.

Given a valid orientation $\widehat{\mathcal{O}}$ of G, we say that $\widehat{\mathcal{O}}$ is *under the restriction* of the bag state s_i of the bag X_i if $\widehat{\mathcal{O}}$ satisfies the following conditions:

1. the orientation (in $\widehat{\mathcal{O}}$) of an edge $e \in E(G[X_i])$ is the same as $s_i(e)$,
2. for each pair of nodes $(u, v) \in X_i \times X_i$, the type of the dependency paths in $G[Y_i \cup \{u, v\}]$ (in the orientation $\widehat{\mathcal{O}}$) are as given by $s_i(u, v)$, and
3. for each node $u \in X_i$, the number of oriented edges (in the orientation $\widehat{\mathcal{O}}$) between u and Y_i coincides with $s_i^-(u)$ and $s_i^+(u)$.

Let us denote by Δ_i the set of all possible states for the bag X_i. Note that $|\Delta_i| \leq 3^{k(k+1)/2} \cdot 5^{k+1} \cdot 16^{k(k+1)}$; since the number of relevant edges, nodes, and pairs of nodes are less than equal to $k(k+1)/2$, $k+1$, $k(k+1)$, respectively. The dynamic programming algorithm will compute in a bottom-up fashion a mapping $A_i : \Delta_i \to \mathbb{R} \cup \{+\infty\}$. For a bag state $s_i \in \Delta_i$, the value $A_i(s_i)$ is the minimum weight of the origins in an optimal valid orientation $\widehat{\mathcal{O}}$ of G_i under the restriction that the state of nodes, edges, and pairs of nodes in X_i is given by s_i. Before we describe our dynamic programming algorithm, let us see how to detect dependency cycles going through a bag X_i.

Detecting dependency cycles and checking the degree constraints: Let i be a node in the tree-decomposition T, let X_i be the corresponding bag in G, and let $s \in \Delta_i$. We define a procedure called `valid`(X_i, s) that return *true* if and only if an orientation $\widehat{\mathcal{O}}$ of G_i under the restriction of s satisfies the degree constraints in X_i and also has no dependency cycles going thorough X_i. Let's denote by $d_s^-(u)$ and $d_s^+(u)$ the in-degree and out-degree of a node u (respectively) in the orientation of $G[X_i]$ given by the state of edges $s(e)$. The total number of in-coming edges to a node u in the orientation $\widehat{\mathcal{O}}$ of G_i under the restriction of s is $d_s^-(u) + s^-(u)$; the first term is the number of in-coming oriented edges to u in $G[X_i]$ and the second term is the number of oriented edges from Y_i to u. The number of out-going edges from u can also be computed in this way; note that we only need to know if the out-degree is ≥ 2 or not.

- `Degree constraints`: The degree constraints at node $u \in X_i$ is satisfied if
  - $s^-(u) + d_s^-(u) \leq 1$ and $((s^-(u) + d_s^-(u) = 1) \Rightarrow s^+(u) + d_s^+(u) \leq 1)$, and
  - if u is a propagation node and $(s^-(u) + d_s^-(u) = 0)$ then $s^+(u) + d_s^+(u) \leq 1$.

  The first condition says that the in-degree of each node is at most 1 and if the in-degree is 1 then the out-degree should be at most 1. The second condition says that if a propagation node is an origin of the orientation then its out-degree is at most 1; note that a domination node with in-degree 0 has no restriction and it can have any number of out-going edges.

- `Dependency cycles`: Let C be a dependency cycle in G_i going through ℓ nodes from X_i in the order $u_1, u_2, \dots, u_\ell$. Each two consecutive nodes u_j, u_{j+1} in the dependency cycle C that appears in X_i are either connected by an edge from $G[X_i]$ or by a dependency path in $G[Y_i \cup \{u_j, u_{j+1}\}]$. Note that the direction of the edge $\{u_j, u_{j+1}\}$ and the type of dependency paths from u_j to u_{j+1} in $G[Y_i \cup \{u_j, u_{j+1}\}]$ are all stored in the bag state s. By combining the types of dependency paths for each consecutive nodes u_j, u_{j+1} (for $i = 1, \dots, \ell$) we can check if there is a dependency cycle going through $u_1, u_2, \dots, u_\ell$. For example, if $\ell = 3$, $s(u_1, u_2) = DD$, $s(u_2, u_3) = UD$, and $s(\{u_3, u_1\}) = \perp$ then there is a dependency cycle going through u_1, u_2, u_3; note that the combination of these 3 dependency paths yields a dependency cycle. In total, there are $O((k+1)!)$ such dependency cycles going through X_i, and checking any one of them needs $O(k)$ time.

In time $O(k \cdot (k+1)!)$, we can check if there is a dependency cycle going through X_i and also check if the degree constraints are satisfied.

Step 1: (Initialization): In this step for each leaf node i of T, we initialize the mapping A_i as follows. For a given state $s_i \in \Delta_i$, we define $A_i(s_i)$ as $+\infty$ if $\texttt{valid}(X_i, s_i) = \mathit{false}$, there exists a node $v \in X_i$ with $s_i^-(v) + s_i^+(v) \neq 0$, or there exists a pair of nodes $u, v \in X_i$ such that $s_i(u, v) \neq \emptyset$. Otherwise, we define $A_i(s_i)$ as the weight of nodes with in-degree 0 in the orientation given by s_i, i.e.,

$$A_i(s_i) = \mathtt{W}\left(\left\{u \in X_i | d^-_{s_i}(u) = 0\right\}\right).$$

Step 2: (Bottom-Up Computation): After initialization, we visit the nodes in T in a bottom-up fashion and at each bag X_i we compute the mapping A_i corresponding to X_i. The update process depends on the type of T-nodes we are visiting. Let j be a child of node i in T. Informally, we say that a bag state s_j of j is *compatible* with a bag state s_i of i if there exists a valid orientation $\widehat{\mathcal{O}}$ that is under the restriction of both s_i and s_j in the graphs G_i and G_j, respectively. The formal definition of compatibility will be given for each type of nodes separately.

Forget node: Suppose i is a forget node with child j, and $X_j = X_i \cup \{x\}$. Note that G_i and G_j have the same set of nodes, so a valid orientation of G_j is also a valid orientation of G_i. The state of nodes and pairs of nodes can be different in these two graphs due to the node x. Given a bag state s_i for the node i, to compute $A_i(s_i)$ we need to compute the set of bag states, $\Delta(s_i)$, of the node j that are compatible with s_i. A bag state s_j of the node j is *compatible* with s_i if the following conditions are satisfied:

1. for each $v \in X_i$:

- if $\{x, v\} \in E(G[X_j])$ then
  - $s_i^-(v) = s_j^-(v) + 1$ if $s_j(\{x, v\}) = (x, v)$, and otherwise $s_i^-(v) = s_j^-(v)$
  - $s_i^+(v) = \min(2, s_j^+(v) + 1)$ if $s_j(\{x, v\}) = (v, x)$, and otherwise $s_i^+(v) = \min(2, s_j^+(v))$
- if $\{x, v\} \notin E(G[X_j])$ then
$$s_i^-(v) = s_j^-(v), s_i^+(v) = s_j^+(v)$$

The number of oriented edges between v and Y_i might increase from the number of oriented edges between v and Y_j, since $x \in Y_i \setminus Y_j$. The above conditions checks these changes.

2. for each edge $e \in E(G[X_i])$ we have $s_i(e) = s_j(e)$

3. for each pair $(u, v) \in X_i \times X_i$ we have:
$$s_i(u, v) = s_j(u, v) \cup (s_j(u, x) \cup t_{(u,x)}) \otimes (s_j(x, v) \cup t_{(x,v)}),$$
where $t_{(u,x)}, t_{(x,v)} \in \{UU, UD, DU, DD\}$ denote the type of the oriented edge (defined by the orientation in s_j) from u to x and the oriented edge from x to v, respectively. This condition is saying that $s_i(u, v)$ (type of dependency paths between u and v) extends s_j by adding the new type of dependency paths that could be formed through the new node x in $Y_i \setminus Y_j$.

Let $\Delta(s_i)$ be the set of all bag states s_j for X_j that are compatible with the bag state s_i for X_i. The function A_i is computed as follows:
$$A_i(s_i) = \min_{s_j \in \Delta(s_i)} A_j(s_j).$$

If $\Delta(s_i) = \emptyset$ then $A_i(s_i)$ is defined to be $+\infty$. The computation of A_i is correct since G_i and G_j are the same graph, and so a valid orientation under the restriction of s_j is also a valid orientation under the restriction of s_i.

Introduce node: Suppose i is an introduce node with child j, and $X_i = X_j \cup \{x\}$. Given a bag state s_i of the bag X_i, to compute $A_i(s_i)$ we need to compute the set of all bag states, $\Delta(s_i)$, of the node j that are compatible with s_i. If $\texttt{valid}(X_i, s_i) = false$, then we define $\Delta(s_i)$ as $\emptyset$. By the properties of the tree decomposition we know that the newly introduced node x has no neighbor in Y_i. Therefore, if $s_i^-(x) + s_i^+(x) \neq 0$ or there exits $v \in X_j$ such that $s(v, x) \neq \emptyset$ or $s(x, v) \neq \emptyset$, then we define $\Delta(s_i)$ as $\emptyset$. Otherwise $\Delta(s_i)$ contains a bag state s_j of X_j if the following conditions are satisfied:

1. for each $e \in E(G[X_j])$ we have: $s_i(e) = s_j(e)$,

2. for each $v \in X_j$ we have: $s_i^-(v) = s_j^-(v)$ and $s_i^+(v) = s_j^+(v)$, and

3. for each pair of nodes $(u, v) \in X_j \times X_j$ we have: $s_i(u, v) = s_j(u, v)$.

Note that adding the node x to G_j does not change the state of nodes, edges, and pairs of nodes in X_j since x has no neighbor in Y_i. Hence, a valid orientation $\widehat{\mathcal{O}}$ under the restriction of s_i is a valid orientation under the restriction of s_j, if s_j is compatible with s_i. Now, we can compute $A_i(s_i)$ using $\Delta(s_i)$ in the following way:

$$A_i(s_i) = \min_{s_j \in \Delta(s_i)} \big\{A_j(s_j) + \mathtt{W}(x) \cdot (1 - d^-_{s_i}(x)) - \mathtt{W}(\{u \in X_j \cap N(x) : s_i(\{x,u\}) = (x,u)\})\big\}.$$

Note that x should be counted as an origin if and only if $d^- s_i(x) = 0$. In the bag state s_i the newly introduced node x could be an origin and also some of the origins in s_j might not be origins in s_i due to oriented edges from x to them. We need to modify the value of $A_j(s_j)$ as above to compensate for these changes.

Join node: Suppose i is a join node with children j and l; so we have $X_i = X_j = X_l$. We need to compute the set, $\Delta(s_i)$, of "compatible" pairs (s_j, s_l) with s_i, where s_j and s_l are bag states of X_j and X_l, respectively. If $\mathtt{valid}(X_i, s_i) = false$ then we define $\Delta(s_i) = \emptyset$. Otherwise, $\Delta(s_i)$ contains the pair (s_j, s_l) if the following conditions ate satisfied:

1. for each $v \in X_i$ we have:
$$s_i^-(v) = s_j^-(v) + s_l^-(v), s_i^+(v) = \min(2, s_j^+(v) + s_l^+(v))$$
2. for each edge $e \in E(G[X_i])$ we have: $s_i(e) = s_j(e) = s_l(e)$, and
3. for each pair of nodes $(u,v) \in X_i \times X_i$, we have $s_i(u,v) = s_j(u,v) \cup s_l(u,v)$.

From the properties of tree decompositions we know that $Y_j \cap Y_l = \emptyset$ and $Y_i = Y_j \cup Y_l$. Now we can compute $A_i(s_i)$ in the following way. By combining the valid orientations in G_j and G_l we get an orientation of G_i. In the obtained orientation a node $u \in X_i$ that is an origin in the orientation of G_j or G_l might not be an origin anymore, or it might be double counted if it is an origin in both G_j and G_l.

$$A_i(s_i) = \min_{(s_j,s_l) \in \Delta(s_i)} \{A_j(s_j) + A_l(s_l) - \mathtt{W}(B_{s_i}(s_j, s_l))\}$$

where $B_{s_i}(s_j, s_l)$ is defined as follows. The set $B_{s_i}(s_j, s_l)$ contains a node $u \in X_i$ if any one the following conditions is satisfied:

- $d^-_{s_i}(u) + s_i^-(u) = 0$; u is counted twice as an origin,
- $d^-_{s_j}(u) + s_j^-(u) = 0$ and $d^-_{s_l}(u) + s_l^-(u) = 1$; u is an origin in G_j but not in G_l, or
- $d^-_{s_j}(u) + s_j^-(u) = 1$ and $d^-_{s_l}(u) + s_l^-(u) = 0$; u is an origin in G_l but not in G_j.

Step 3: (At root r): Finally, we compute the minimum weight of origins in an optimal valid orientation of G by finding the minimum of $A_r(s)$ over all possible $s \in \Delta_r$.

The bottle neck in the computation of the mapping A_i is at a join node i, where for each bag state s_i we need to consider all pairs of compatible bag states of its two children. Hence, at each bag X_i we can compute A_i in time $O\left(\left(3^{k(k+1)/2} \cdot 5^{k+1} \cdot 16^{k(k+1)}\right)^3\right) = O(c^{k^2})$, for some constant c. Therefore, the total running time of our algorithm is $O(c^{k^2} \cdot n)$.

B.2 The ℓ-round general propagation problem

Our dynamic programming algorithm is based on valid timed-orientations. This dynamic programming algorithm is similar to the dynamic programming algorithm for PDS given in [30].

Fix a parameter ℓ and consider the ℓ-round General Propagation problem. Assume that the graph $G = (V, E)$ and $V' \subseteq V$ and a nice tree decomposition $\langle \{X_i \subseteq V \mid i \in I\}, T = (I, F)\rangle$ of G with tree-width k are given as input. We start by describing the state of bags in our dynamic programming algorithm.

The state of a bag: Given a valid timed-orientation of G, the state of a bag X_i describes the orientation of the edges in $G[X_i]$ and the time label assigned to each node. In order to check the conditions of the valid timed-orientation, we need to store more information.

- `State of an edge`: The state of an edge $e = \{u, v\} \in E(G[X_i])$ denoted by $s(e)$ shows the orientation that is assigned to e; i.e., $s(e) \in \{(u, v), (v, u), \perp\}$
- `State of a node`: For each node $v \in X_i$ we define the following five states:
  - $s^-(v)$: denotes the number of in-coming edges to v from Y_i,
  - $s^+(v)$: denotes the number of out-going edges from v to Y_i,
  - $s^t(v)$: denotes the time label assigned to v in the orientation,
  - $s^m(v)$: denotes the maximum time label assigned to the neighbors of v in Y_i except it's unique out-neighbor (if there is any),
  - $s^r(v)$ denotes the time label assigned to the out-neighbor of v in Y_i (if there is any). Informally speaking, this shows the round in which v covers it's unique out-neighbor by an application of the propagation rule.

For each node $v \in X_i$, $s^t(v)$ takes a value from $\{0, 1, 2, \ldots, \ell\} \cup \{+\infty\}$, $s^-(v)$ takes a value from $\{0, 1\}$, $s^+(v)$ takes a value from $\{0, 1, 2\}$, $s^m(v)$ takes a value from $\{0, 1, \ldots, \ell\} \cup \{+\infty\}$, and $s^r(v)$ takes a value from $\{0, 1, 2, \ldots, \ell\} \cup \{+\infty\}$. The $s^+(v) = 2$ denotes that there are at least 2 oriented edges from v to Y_i. The state

$s^r(v)$ is needed to check if the property (P5) in the valid time-orientation is satisfied. If v is a source node of domination type or it has no out-neighbor in the orientation then $s^r(v)$ is not used and has value $s^r(v) = 0$ otherwise it keeps the time label assigned to it's unique out-neighbor (in Y_i).

Let us denote by Δ_i the set of all possible states of the bag X_i. It is straightforward to check that the number of bag states for X_i is $|\Delta_i| \leq 3^{m_i} \times 5^{n_i} \times (\ell + 2)^{3n_i}$. For each bag X_i we will compute and store a mapping $A_i : \Delta_i \to \mathbb{R} \cup \{+\infty\}$. For a bag state $s \in \Delta_i$, the value $A_i(s)$ shows the weight of origins in the optimal valid timed-orientation of the subproblem induced on G_i under the restriction that the orientation of edges and labeling of nodes in X_i is defined by the state s.

We define a procedure called `valid`(X_i, s) that returns *true* if the timed orientation of $G[X_i]$ given by the state s satisfies the properties (P1-P5) in the definition of the valid timed-orientation. Note that the number of oriented edges between a node v to Y_i are stored in the state s. Also using the direction of edges in $G[X_i]$ given by the state s, we can compute the in-degree of v in $G[X_i]$, denoted by $d_s^-(u)$, and the out-degree of v in $G[X_i]$, denoted by $d_s^+(v)$. After computing these information, it is straightforward to check the properties (P1,P3,P4). Property (P5) should be checked for each directed edge (u, v). The important case to check is when u is not a source node of domination type; note that this is the only case that the propagation rule applies. In this case, we only need to check property (P5) when all neighbors of u are in G_i and u has no neighbors in $G \setminus G_i$, and this happen either at the root node or at a Forget Node. In order to do this test, we need to keep the time label assigned to v in the state $s^r(u)$ since v might not be in the considered bag and could be deep down in the tree, and also note that $s^m(u)$ should be the maximum of time labels of all neighbors of u except v. Note that property (P2) should also be checked at a Forget Node. Observe that, the `valid`(X_i, s) procedure can be done in time $O(k^2)$. Now we describe our dynamic programming algorithm.

Step 1: (Initialization) In this step for each leaf node i in the tree T, we define (initialize) the mapping A_i for each $s \in \Delta_i$. If `valid`$(X_i, s) =$ false or there exists a node v such that one of $s^-(v), s^+(v), s^m(v), s^r(v)$ is not 0, then define $A_i(s) = +\infty$. Otherwise, let $A_i(s)$ be the weight of nodes with $s^t(v) = 0$.

Step 2: (Bottom-Up Computation) After initialization, we visit the nodes of T in a bottom-up fashion and at each bag X_i we compute the mapping A_i corresponding to X_i. The update process depends on the type of T-nodes we are visiting. Recall that there are three node types: Forget Node, Join Node, Introduce Node. In the following, we describe how to compute A_i for each of these three node types.

Forget Node: Suppose i is a forget node with child j, and assume that $X_j = X_i \cup \{x\}$. The bag states $s_i \in \Delta_i$ and $s_j \in \Delta_j$ are called *forget-compatible* and denoted by $s_i \overset{F}{\sim} s_j$, if

(F1) $\forall e \in E(G[X_j]) : s_i(e) = s_j(e)$.

(F2) $\forall v \in V(G[X_j])$:

- $s_i^-(v) = s_j^-(v) + 1$ if $s_j(\{x,v\}) = (x,v)$, otherwise $s_i^-(v) = s_j^-(v)$
- $s_i^+(v) = \min(2, s_j^+(v) + 1)$ if $s_j(\{x,v\}) = (v,x)$, otherwise $s_i^+(v) = \min(2, s_j^+(v))$
- $s_i^t(v) = s_j^t(v)$.

(F3) $\forall v \in V(G[X_j])$:

$$s_i^m(v) = \begin{cases} \max\{s_j^m(v), s_j^t(x)\} & \text{if } \{x,v\} \in E(G[X_j]) \text{ and } s_j(\{x,v\}) \neq (v,x) \\ s_j^m(v) & \text{otherwise} \end{cases}$$

(F4) The property (P5) should be satisfied on x if x has an out-going edge in G_i. Also if there is a directed edge from a node v to x and v is not a source node of domination type, then $s_i^r(v)$ must be equal to $s_j^t(x)$. Also property (P2) should be satisfied on x.

Now we compute the mapping A_i for the bag X_i as follows: $\forall s \in \Delta_i$

$$A_i(s) = \begin{cases} +\infty & \texttt{valid}(X_i, s) = \text{false} \\ \min\{A_j(s_j) : s_j \in \Delta_j, s \overset{F}{\sim} s_j\} & \text{otherwise} \end{cases}$$

Introduce Node: Suppose i is an introduce node with child j, and assume that $X_i = X_j \cup \{x\}$. The bag states $s_i \in \Delta_i$ and $s_j \in \Delta_j$ are called *introduce-compatible* and denoted by $s_i \overset{I}{\sim} s_j$, if

(I1) $\forall e \in E(G[X_j]) : s_i(e) = s_j(e)$

(I2) $\forall v \in V(G[X_j]) : s_j^t(v) = s_i^t(v), s_j^-(v) = s_i^-(v), s_j^+(v) = s_i^+(v)$, $s_j^m(v) = s_i^m(v)$, and $s_j^r(v) = s_i^r(v)$.

We now compute the mapping A_i for the bag X_i as follows. If $\texttt{valid}(X_i, s_i) = \text{false}$, $s_i^r(x) \neq 0$, or $s_i^m(x) \neq 0$, then $A_i(s_i) = +\infty$. Otherwise,

$$A_i(s_i) = \min_{s_i \overset{I}{\sim} s_j} \left\{ A_j(s_j) + \left\{ \begin{array}{cc} \texttt{W}(x) & \text{if } s_i^t(x) = 0 \\ 0 & \textit{otherwise} \end{array} \right\} \right.$$

Join Node: Suppose that i is a join node with j and k as its children, and assume that $X_i = X_j = X_k$. We say $s_j \in \Delta_j$ and $s_k \in \Delta_k$ are *join-compatible* with $s_i \in \Delta_i$ and denote it by $s_i \overset{J}{\sim} (s_j, s_k)$, if the following conditions hold:

(J1) $\forall v \in V(GX_i])$:

- $s_i^-(v) = s_j^-(v) + s_k^-(v)$

- $s_i^+(v) = \min(2, s_j^+(v) + s_k^+(v))$
- $s_i^m(v) = \max(s_j^m(v), s_k^m(v))$
- $s_i^t(v) = s_j^t(v) = s_k^t(v)$
- $s_l^r(v) = \max(s_i^r(v), s_j^r(v))$.

(J2) $\forall e \in E(G[X_i]) : s_i(e) = s_j(e) = s_k(e)$

We now compute the mapping A_i for the bag X_i as follows. If $\texttt{valid}(X_i, s_i) = \text{false}$ then $A_i(s_i) = +\infty$. Otherwise,

$$A_i(s_i) = \min_{s_i \overset{J}{\sim} (s_j, s_k)} \left\{ A_j(s_j) + A_k(s_k) - \mathtt{W}\left(\left\{v \in X_i : s_i^t(v) = 0\right\}\right)\right\}$$

Step 3: (At root r) Let r be the root of the tree decomposition T. Finally we compute the minimum weight of origins in the optimal solution for ℓ-round General Propagation problem in the following way:

$$\delta(G) = \min\{A_r(s) : s \in \Delta_r\}.$$

Note that at the root node, we need to check properties (P2) and (P5). If (P2) or (P5) is not satisfied in a bag state s, then $A_r(s) = \infty$. Let m_e be the maximum number of edges that a bag can have, and also note that the maximum number of nodes in a bag is $(k + 1)$. It is easy to check that each step of the dynamic programming algorithm can be computed in time $O(c^{m_e + k \log \ell})$, for some constant c. This shows that the total running time of our algorithm is $O(c^{m_e + k \log \ell} \cdot |V|)$.

B.3 The directed PDS problem

In this section, we describe our dynamic programming algorithm for the Directed PDS problem. This algorithm is similar to the dynamic programming algorithm designed by Guo et al. [30] to optimally solve PDS for undirected graphs of bounded tree-width.

Consider a valid coloring $\mathcal{C}$ of the digraph G. We store the color of the edges in each bag by assigning a state to that bag (the formal definition of a state will follow). We can reconstruct the coloring $\mathcal{C}$ from the states of all bags in the tree decomposition of G; so there is no need to store the coloring $\mathcal{C}$ in the dynamic programming algorithm. This algorithm is similar to the dynamic programming algorithm for the General Propagation problem. The Blue edges in the coloring correspond to the unoriented edges in the valid orientations, and the Red edges correspond to the oriented edges. Here, we only describe the states of a bag. The rest of the dynamic programming algorithm can be easily adapted for the Directed PDS problem.

The state of a bag: Given a coloring $\mathcal{C}$, the state of a bag X_i describes the coloring of the edges in $G[X_i]$. In order to detect the dependency cycles in the coloring $\mathcal{C}$ without reconstructing the whole coloring, we need to store some more information. This extra information enables us to detect a dependency cycle in G_i which goes through X_i, by considering only the state of the bag X_i. A *bag state* s contains the following information:1) state of each edge, 2) state of each node, and 3) state of each pair of nodes for $G[X_i]$.

- `State of an edge`: The state of an edge $e \in E(G[X_i])$ denoted by $s(e)$ is the color that is assigned to e; i.e., $s(e) \in \{R, B\}$.
- `State of a node`: The state of a node $v \in X_i$ has two parameters $s^-(v)$ and $s^+(v)$, where $s^-(v)$ denotes the number of in-coming red edges to v from nodes in Y_i, and $s^+(v)$ denotes the number of out-going red edges to Y_i from v. The values that these two parameters can take is the same as in the dynamic programming algorithm for the General Propagation problem.
- `State of a pair of nodes`: A *dependency path* from u to v is a path P where all red edges in P are directed from u to v and all blue edges are directed from v to u. We categorize dependency paths according to the color of their first and last edges. There are 4 possible types RR, RB, BR, BB; for example, a path of type RB is a path with the first edge colored red and the last edge colored blue. For a pair $(u, v) \in X_i \times X_i$ $(u \neq v)$ the state of (u, v) denoted by $s(u, v)$ shows the type of dependency paths from u to v in $G[Y_i \cup \{u, v\}]$; that is, $s(u, v) \subseteq \{RR, RB, BR, BB\}$. Note that there are $2^4 = 16$ different states for each pair of nodes; $s(u, v) = \emptyset$ means that there is no dependency path from u to v in $G[Y_i \cup \{u, v\}]$.

B.4 The target set selection problem

The dynamic programming algorithm for the Target Set Selection is similar to the dynamic programming algorithm for the general Propagation problem. Here, we only describe the state of bags needed for our algorithm. The other modifications are straightforward, and we skip them here.

The state of a bag: Given a valid target-orientation $\widehat{\mathcal{O}}$ of G, the state of a bag X_i describes the orientation of the edges in $G[X_i]$. In order to detect the directed cycles in $\widehat{\mathcal{O}}$ without reconstructing the whole orientation, we need to store some more information. This extra information enables us to detect a directed cycle in G_i that goes through X_i, by considering only the state of the bag X_i. A *bag state* s contains the following information: 1) state of each edge, 2) state of each node, and 3) state of each pair of nodes in $G[X_i]$.

- `State of an edge`: The state of an edge $e = \{u, v\} \in E(G[X_i])$ denoted by $s(e)$ shows the orientation that is assigned to e; i.e., $s(e) \in \{(u, v), (v, u), \perp\}$.
- `State of a node`: For each node $v \in X_i$ we define a state denoted by by $s^-(v)$ that is the number of in-coming edges to v from Y_i.

- `State of a pair of nodes`: For a pair $(u, v) \in X_i \times X_i$ $(u \neq v)$ the state of (u, v) denoted by $s(u, v)$ shows the existence of a directed path from u to v in $G[Y_i \cup \{u, v\}]$. $s(u, v) = \text{true}$ if there is such a directed path from u to v, and it is assigned false otherwise.

Let T be the maximum threshold assigned to the nodes of G. It is easy to check that there are at most $3^{k(k+1)/2} \cdot T^{(k+1)} \cdot 2^{k(k+1)}$ bag states, where k is the tree-width of G. Note that since the largest threshold is T, each node has at most T different values for $s^-(v)$. Hence, the total running time of our algorithm is $O(c^{k^2} \cdot T^{k+1} \cdot |V(G)|)$ for some constant c.

B.5 The ℓ-round target set selection problem

The dynamic programming algorithm for the ℓ-round Target Set Selection is similar to the dynamic programming algorithm for the ℓ-round General Propagation problem. Here, we only describe the state of bags needed for our algorithm. The other modifications are straightforward, and we skip them here.

The state of a bag:

- `State of an edge`: The state of an edge $e = \{u, v\} \in E(G[X_i])$ denoted by $s(e)$ shows the orientation that is assigned to e.
- `State of a node`: For each node $v \in X_i$ we define the following states:
  - $s^-(v)$ denotes the number of in-coming edges from Y_i to v
  - $s^+(v)$ denotes the number of out-going edges from v to Y_i; note that for the out-degree we only need to know if $s^+(v) > 0$, so it takes value from $\{0, 1\}$.
  - $s^r(v)$ denotes the time-label assigned to v.
  - $s^m(v)$ denotes the maximum time-label assigned to the in-neighbors of v in Y_i.

Let T be the maximum threshold assigned to the nodes of G. It is easy to check that there are at most $3^{k(k+1)/2} \cdot (2T)^{(k+1)} \cdot (\ell+2)^{2k+2}$ bag states, where k is the tree-width of G. The total running time of our algorithm is $O(c^{k^2+k\log\ell} \cdot T^{k+1} \cdot |V(G)|)$ for some constant c.

Bibliography

[1] A. Aazami. Domination in graphs with bounded propagation: algorithms, formulations and hardness results. *accepted for publication in Journal of Combinatorial Optimization*, July 2008. 1

[2] A. Aazami, J. Cheriyan, and K. Raju Jampani. Hardness results and approxiamtion algorithms for packing Steiner trees. *manuscript, in preparation*, Oct. 2008. 1

[3] A. Aazami and M. D. Stilp. Approximation algorithms and hardness for domination with propagation. *22 pages, submitted to a journal on Aug. 2006 (revision submitted on Oct 2007).* 1

[4] A. Aazami and M. D. Stilp. Approximation algorithms and hardness for domination with propagation. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, volume 4627 of *LNCS*, pages 1–15. Springer, 2007. 1, 8

[5] J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica*, 33(4):461–493, 2002. 31, 38

[6] N. Alon. A propagation process on Cayley graphs. 2008. 8

[7] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. ACM*, 41(1):153–180, 1994. 2, 46, 47, 48, 49, 58, 59

[8] T. L. Baldwin, L. Mili, M. B. Boisen, and R. Adapa. Power system observability with minimal phasor measurement placement. *IEEE Transactions on Power Systems*, 8(2):707–715, 1993. 7, 30

[9] M. Bellare, S. Goldwasser, C. Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 294–304, 1993. 25

[10] N. Berger, C. Borgs, J. T. Chayes, and A. Saberi. On the spread of viruses on the internet. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 301–310, 2005. 8

[11] H. L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36:116–126, 1988. 59, 61

[12] Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995. 89

[13] D. J. Brueni. Minimal PMU placement for graph observability, a decomposition approach. *M.S. thesis*, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1993. 7, 30

[14] D. J. Brueni and L. S. Heath. The PMU placement problem. *SIAM J. Discret. Math.*, 19(3):744–761, 2005. 31

[15] D. Burgarth and V. Giovannetti. Full control by locally induced relaxation. *Physical Review Letters*, 99:100501, 2007. 8

[16] N. Chen. On the approximability of influence in social networks. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1029–1037, 2008. 8, 64

[17] J. Cheriyan and M.R. Salavatipour. Hardness and approximation results for packing steiner trees. *Algorithmica*, 45(1):21–43, 2006. 70

[18] J. Cheriyan and M.R. Salavatipour. Packing element-disjoint steiner trees. *ACM Transactions on Algorithms*, 3(4), 2007. 70, 71

[19] J. Chuzhoy and S. Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 179–188, 2007 (Full version is avaiable on the home pages of both authors). 23, 25

[20] V. Chvatal. A greedy heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979. 30

[21] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique width. In *Workshop on Graph-Theoretic Concepts in Computer Science*, pages 1–16, 1998. 30, 48

[22] E. D. Demaine and M. T. Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 590–601, 2005. 46, 47, 48

[23] R. Diestel. *Graph Theory.* Springer-Verlag, New York, 2nd edition, 2000. 4, 12, 37, 74, 88

[24] M. Dorfling and M. A. Henning. A note on power domination in grid graphs. *Discrete Applied Mathematics*, 154(6):1023–1027, 2006. 31, 39

[25] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998. 30

[26] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980. 77

[27] A. Frank, T. Király, and M. Kriesell. On decomposing a hypergraph into k connected sub-hypergraphs. *Discrete Applied Mathematics*, 131(2):373–383, 2003. 70, 73, 75

[28] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY, USA, 1979. 4, 31

[29] AIM Minimum Rank Special Graphs Work Group. Zero forcing sets and the minimum rank of graphs. *Linear Algebra and its Applications*, 428(7):1628–1648, 2008. 7, 8, 12, 28

[30] J. Guo, R. Niedermeier, and D. Raible. Improved algorithms and complexity results for power domination in graphs. In *Proceedings of the 15th International Symposium on Fundamentals of Computation Theory*, volume 3623 of *LNCS*, pages 172–184. Springer, 2005 (to appear in Algorithmica). 2, 31, 38, 48, 49, 90, 96, 99

[31] T. W. Haynes, S. M. Hedetniemi, S. T. Hedetniemi, and M. A. Henning. Domination in graphs applied to electric power networks. *SIAM J. Discrete Math.*, 15(4):519–529, 2002. 30

[32] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Domination in Graphs: Advanced Topics*. Marcel Dekker, 1998. 40

[33] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, 1998. 40

[34] H.R. Hind and O. Oellermann. Menger-type results for three or more vertices. *Congressus Numerantium*, 113:179–204, 1996. 71, 72

[35] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. In *Proceedings of the 18th Annual ACM Symposium on Discrete Algorithms*, pages 637–644, Philadelphia, PA, USA, 2007. 62

[36] K. Jain, M. Mahdian, and M.R. Salavatipour. Packing steiner trees. In *SODA*, pages 266–274, 2003. 70

[37] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974. 30

[38] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *J. Comb. Theory, Ser. B*, 82(1):138–154, 2001. 62

[39] P. Kaski. Packing steiner trees with identical terminal sets. *Inf. Process. Lett.*, 91(1):1–5, 2004. 70

[40] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003. 8, 64

[41] D. Kempe, J. Kleinberg, and E. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP '05: Automata, Languages and Programming, 32nd International Colloquium*, pages 1127–1138, 2005. 8, 64

[42] T. Kloks. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994. 38, 89

[43] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. Parameterized power domination complexity. *Inf. Process. Lett.*, 98(4):145–149, 2006. 30, 48

[44] G. Kortsarz. On the hardness of approximating spanners. *Algorithmica*, 30(3):432–450, 2001. 19, 20, 32

[45] M. Kriesell. Edge-disjoint trees containing some given vertices in a graph. *J. Comb. Theory, Ser. B*, 88(1):53–65, 2003. 70

[46] L.C. Lau. *On approximate min-max theorems for graph connectivity problems*. PhD thesis, University of Toronto, 2006. 70

[47] L.C. Lau. An approximate max-steiner-tree-packing min-steiner-cut theorem. *Combinatorica*, 27(1):71–90, 2007. 70

[48] C. S. Liao and D. T. Lee. Power domination problem in graphs. In *Proceedings of the 11th International Computing and Combinatorics Conference*, volume 3595 of *LNCS*, pages 818–828. Springer, 2005. 31

[49] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Mathematics*, 13:383–390, 1975. 30

[50] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994. 30

[51] K. Menger. Zur allegemeinen kurventheorie. *Fund. Math.*, 36:95–115, 1927. 69

[52] M. Middendorf. Minimum broadcast time is NP-complete for 3-regular planar graphs and deadline 2. *Inf. Process. Lett.*, 46(6):281–287, 1993. 8

[53] M. Middendorf and F. Pfeiffer. On the complexity of the disjoint paths problem. *Combinatorica*, 13(1):97–107, 1993. 76

[54] L. Mili, T.L. Baldwin, and A.G. Phadke. Phasor measurements for voltage and transient stability monitoring and control. In *Proceedings of the EPRI-NSF Workshop on Application of Advanced Mathematics to Power Systems*, 1991. 7, 30

[55] S. Morris. Contagion. *Review of Economic Studies*, 67(1):57–78, 2000. 8

[56] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 128–134, New York, NY, USA, 2007. ACM. 8

[57] C. S. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of London Mathemathical Society*, 36:445–450, 1961. 70

[58] G. Naves. The hardness of routing two pairs on one face. Technical report, Laboratoire G-SCOP, avaiable at http://hal.archives-ouvertes.fr/hal-00313944/en/, 2008. 71, 81

[59] G. Naves and A. Sebő. Multiflow feasibility: an annotated tableau. In *Research Trends in Combinatorial Optimization*, pages 261–283. Springer Berlin Heidelberg, 2008. 71, 76, 81

[60] J. Obdrzálek. Dag-width: connectivity measure for directed graphs. In *Proceedings of the 17th Annual ACM Symposium on Discrete Algorithms*, pages 814–821. ACM Press, 2006. 62

[61] R. Pastor-Satorras and A. Vespignani. Epidemics and immunization in scale-free networks. In *Handbook of Graphs and Networks: From the Genome to the Internet, edited by S. Bornholdt, H. G. Schuster*, pages 111–130. Wiley-VCH, 2003. 8

[62] D. Peleg. Local majority voting, small coalitions and controlling monopolies in graphs: A review. In *Proceedings of the 3rd Colloquium on Structural Information & Communication Complexity*, pages 170–179, 1996. 8

[63] J. Plesník. The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Inf. Process. Lett.*, 8(4):199–201, 1979. 14

[64] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002. 8

[65] F. S. Roberts. Challenges for discrete mathematics and theoretical computer science in the defense against bioterrorism. In *Mathematical and Modeling Approaches in Homeland Security, SIAM Frontiers in Applied Mathematics Series, edited by H. T. Banks and C. Castillo-Chavez*, pages 1–34, 2003. 8

[66] N. Robertson and P.D. Seymour. Graph minors: X. obstructions to tree-decomposition. *J. Comb. Theory Ser. B*, 52(2):153–190, 1991. 13

[67] N. Robertson, P.D. Seymour, and R. Thomas. Quickly excluding a planar graph. *J. Comb. Theory, Ser. B*, 62(2):323–348, 1994. 38

[68] S. Severini. Nondiscriminatory propagation on trees. *to appear in Journal of Physics A: Math. Theor. (Fast Track Communication)*, 2008. 8

[69] P. D. Seymour and R. Thomas. Call routing and the ratcatcher. *Combinatorica*, 14(2):217–241, 1994. 89

[70] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 435–441, New York, NY, USA, 1996. ACM Press. 30

[71] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal of London Mathemathical Society*, 36:221–230, 1961. 70

[72] V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001. 4, 34