Journal of Artificial Intelligence Research 49 (2014) 569-600 Submitted 04/2013; published 04/2014

Inapproximability of Treewidth, One-Shot Pebbling, and Related Layout Problems

Yu (Ledell) Wu WUYU@CS.TORONTO.EDU
Per Austrin AUSTRIN@CS.TORONTO.EDU
Toniann Pitassi TONI@CS.TORONTO.EDU
David Liu LIUDAVID@CS.TORONTO.EDU
Department of Computer Science
University of Toronto, Ontario, Canada

Abstract

Graphical models, such as Bayesian Networks and Markov networks play an important role in artificial intelligence and machine learning. Inference is a central problem to be solved on these networks. This, and other problems on these graph models are often known to be hard to solve in general, but tractable on graphs with bounded Treewidth. Therefore, finding or approximating the Treewidth of a graph is a fundamental problem related to inference in graphical models. In this paper, we study the approximability of a number of graph problems: Treewidth and Pathwidth of graphs, Minimum Fill-In, One-Shot Black (and Black-White) pebbling costs of directed acyclic graphs, and a variety of different graph layout problems such as Minimum Cut Linear Arrangement and Interval Graph Completion. We show that, assuming the recently introduced Small Set Expansion Conjecture, all of these problems are NP-hard to approximate to within any constant factor in polynomial time.

1. Introduction

Graphical models provide a computational framework for efficiently manipulating probability distributions over high dimensional spaces, often involving hundreds of thousands of variables. This framework has found applications in an enormous range of domains including: medical and fault diagnosis, image understanding, speech recognition, web search, coding theory, and statistical physics (Koller & Friedman, 2009). A graphical model is an efficient representation of a joint distribution over some set of n random variables. Even if the random variables are binary, it is well known that an arbitrary joint distribution requires the specification of 2^n probabilities. Luckily, in the real world, there is often structure in the distribution that allows one to express it more succinctly. A graphical model represents such a joint probability distribution by a graph where the vertices represent the random variables, and the dependences are modeled by the graph structure. Associated with each vertex of the graph is a conditional probability table, which specifies the conditional probabilities of this random variable, conditioned on its neighboring vertices. The two most common types of graphical models are Bayesian networks (also called belief networks), where the underlying graph is directed, and Markov networks (also called Markov random fields), where the underlying graph is undirected. The most basic problem in graphical models is the *inference problem*, which is the problem of computing the posterior marginal distribution of a variable at some vertex. Unfortunately, inference in general is well-known to be NP-hard to compute exactly as well as to approximate (Roth, 1996).

Despite this intractability, an important class of *bounded Treewidth* instances of probabilistic inference has been identified and shown to be exactly computable in polynomial time. The Treewidth of a graph (Robertson & Seymour, 1984, 1986) is a fundamental parameter of a graph that measures how close the graph is to being a tree. Treewidth is very closely related to the other notions in machine learning such as Branch-width, Clique-width and Elimination-width (for an overview of Treewidth and related notions, see Bodlaender, Gilbert, Hafsteinsson, & Kloks, 1995). On graphs with small Treewidth and where the tree decomposition is known, a dynamic programming algorithm yields a polynomial-time algorithm. Particular algorithms for probabilistic inference on bounded Treewidth graphs are the junction-tree method, variable elimination and clique trees (e.g. see Koller & Friedman, 2009, ch. 9, 10). These algorithms runs in time exponential in the Treewidth of the tree decomposition and polynomial in the size of the graph. Thus for graphs where a tree decomposition of bounded Treewidth is given, inference is polynomial-time computable.

The same ideas also yield polynomial-time algorithms and often even linear time algorithms for small Treewidth instances for an astonishing variety of other NP-hard problems, including: satisfiability, counting satisfying assignments, constraint satisfaction, vertex cover, maximum independent set, Hamiltonian circuit, query optimization, matrix decomposition, and more generally all problems definable in monadic second-order logic. (See the excellent survey Bodlaender, 2005 for motivation, including theoretical as well as practical applications of Treewidth.) One catch is that for all of these problems, the algorithm must begin by finding a tree decomposition, and then use the decomposition to solve the problem. Given the tree decomposition, the algorithm is typically exponential in the width of the underlying tree decomposition. Thus there is a need for efficient algorithms to actually compute the Treewidth of a given graph, and to find tree decompositions with optimal or close to optimal width.

Unfortunately, while there are many good heuristics for finding a good tree decomposition, it is NP-hard in general to determine the Treewidth of a graph (Arnborg, Corneil, & Proskurowski, 1987). However, Bodlaender et al. (1995) obtained an $O(\log n)$ factor approximation algorithm for Treewidth. In fact, they actually show that if there is a factor c approximation algorithm for vertex separator, then there is an $O(c)$ approximation algorithm for Treewidth. And if there is a factor b approximation algorithm for Treewidth then there is an $O(b \log n)$ approximation algorithm for the related Pathwidth problem. The best currently known approximation factor for vertex separator is $O(\sqrt{\log n})$ (Feige, Hajiaghayi, & Lee, 2005) and thus the best algorithm for Treewidth finds a tree decomposition that is within an $O(\sqrt{\log n})$ factor of the optimal width, and an $O((\sqrt{\log n})(\log n))$ factor approximation algorithm for Pathwidth.

It is a longstanding open question whether or not there is a *constant* factor approximation algorithm for Treewidth. Such an algorithm would lead to faster algorithms to find good tree-decompositions for all of the problems mentioned above. The current best known algorithm that achieves a constant factor approximation for Treewidth runs in time $2^{O(w)}O(n)$, where w is the Treewidth of the underlying graph, and achieves a factor 5 approximation (Bodlaender, 2007) (Earlier approximation algorithms with similar runtimes are Reed, 1992; Amir, 2001). In a book devoted to Treewidth Kloks (1994, p. 62) states:

> "We feel this is one of the biggest open problems in the research dealing with Treewidth and Pathwidth at the moment. If the fast algorithms for solving NP-hard problems for graphs with bounded Treewidth are ever to become of practical importance, it is undoubtedly of importance to find good tree-decompositions for these graphs. Since the (current best) approximations do not make many of these algorithms practical, it is of great interest to know whether approximations with a small constant exist."

Nearly twenty years later, it is still an open problem whether or not there is a polynomial-time algorithm to approximate Treewidth to within a constant factor. Similarly, the approximability of many related graph layout problems is also unresolved, including Minimum Cut Linear Arrangement and Interval Graph Completion. In this paper, we make an important step to resolve this problem by showing that Treewidth, Pathwidth, and a host of related graph layout problems are hard to approximate to within any constant factor, under the Small Set Expansion (SSE) conjecture (Raghavendra & Steurer, 2010).

The SSE conjecture is a strengthened version of the conjecture that P is different from NP and warrants some explanation. In the next subsection (Section 1.1), we explain the SSE conjecture, and how it relates to the P versus NP question and to related conjectures. We then state our main hardness results for Treewidth, pebbling problems and graph layout problems (Sections 1.2, 1.4, and 1.5), and discuss related results in Section 1.6.

1.1 The Small Set Expansion Conjecture

The P versus NP problem is the most important and intriguing open problem in the field of computational complexity theory. Many decision problems in theory and practice have been proven to be NP-hard, which indicates that they are impossible to compute in polynomial time, under the widely believed conjecture that P $\neq$ NP. The discovery of the PCP theorem in the late 80's (Arora, Lund, Motwani, Sudan, & Szegedy, 1998; Arora & Safra, 1998) made it possible to prove that for many optimization problems, approximating the optimal value to within a certain factor is as hard as computing the exact optimal value. In other words, under the conjecture that P $\neq$ NP, it is not possible to approximate certain optimization problems within some factor that depends on the problem. Celebrated results show that it is NP-hard to approximate MAX-3SAT within a ratio of $\frac{7}{8} + \epsilon$ for any $\epsilon > 0$ (Håstad, 2001), which gives the optimal lower bound, since there is a simple algorithm that achieves an approximation ratio of $\frac{7}{8}$. Also, it is NP-hard to approximate clique to within a $n^{1-\epsilon}$ factor for any $\epsilon > 0$ (Håstad, 1999). Despite this success, for many important problems, the hardness of approximation results obtained through the PCP theorem have not matched the best approximation algorithms known. For example, there are still significant gaps in our understanding of the optimal approximability factor for important problems such as: Vertex Cover, Max-Cut, Bipartite Clique, and Kernel Clustering.

The formulation of the Unique Games Conjecture (UGC) due to Khot (2002) was intended to clarify the approximability of many optimization problems. The conjecture postulates that the problem of determining the value of a certain type of game, known as a unique game, is NP-hard. The conjecture has inspired a remarkable body of work since its formulation. Under UGC, many of the known algorithms in approximation are proven to be tight (for an excellent survey on this topic, see Khot & Vishnoi, 2005). For instance, under the UGC, the Vertex Cover problem is NP-hard to approximate within a factor of $2 - \epsilon$, for any $\epsilon > 0$ (Khot & Regev, 2008). Perhaps most strikingly, Raghavendra (2008) proved that under the UGC, the semi-definite programming (SDP) approximation algorithm for a large class of constraint satisfaction problems (CSP) are essentially the best one can hope for. More specifically, he showed that every maximum constraint satisfaction problem (Max CSP) has an associated sharp approximation threshold τ: for every $\epsilon > 0$, one can achieve a $\tau - \epsilon$ approximation in polynomial time using SDP, but obtaining a $\tau + \epsilon$ approximation is NP-hard. Thus, the UGC has become the central open problem in inapproximability and encapsulates the barrier of designing better polynomial time approximation algorithms for a large class of problems.

Despite this tremendous progress, still there remain important yet stubborn problems such as Treewidth, Balanced Separator, Minimum Linear Arrangement (MLA), and many other graph layout problems whose approximation status remains unresolved even assuming the UGC. Intuitively this is because the hard instances for these problems seem to require a certain global structure such as expansion. (Expansion is a graph property that is akin to high connectivity, and requires that every subset of vertices that is not too large has a large boundary.) Typical reductions for these problems are gadget reductions which preserve global properties of the unique games instance, such as the lack of expansion. Therefore, barring radically new types of reductions that do not preserve global properties, proving hardness for these problems seems to require a stronger version of UGC, where the instance is guaranteed to have certain expansion properties.

In the work of Raghavendra and Steurer (2010), the Small Set Expansion (SSE) Conjecture was introduced, and it was shown that it implies the UGC, and that the SSE Conjecture follows if one assumes that the UGC is true for somewhat expanding graphs. In follow-up work by Raghavendra et al. (2012), it was shown that the SSE Conjecture is in fact equivalent to the UGC on somewhat expanding graphs, and that the SSE Conjecture implies NP-hardness of approximation for balanced separator and MLA. In this light, the Small Set Expansion conjecture serves as a natural unified conjecture that yields all of the implications of UGC and also hardness for expansion-like problems that could not be resolved with the UGC.

Our main contribution in this paper is to prove that a wide range of other graph layout problems are SSE-hard to approximate to within any constant factor. For these problems, no evidence of hardness of approximation was known prior to our results. Moreover, we show that Treewidth, Pathwidth, and Minimum Fill-In are SSE-hard to approximate within any constant factor. This is the first result giving hardness of (relative) approximation for these problems, and gives evidence that no constant factor approximation algorithm exists for them.

It should be noted that the status of the SSE conjecture is very open at this point. In particular, recent results (Arora, Barak, & Steurer, 2010; Barak, Raghavendra, & Steurer, 2011; Guruswami & Sinop, 2011) give subexponential-time algorithms for small set expansion. Still despite this recent progress providing evidence against the SSE conjecture, it remains open. Our SSE-hardness results for Treewidth and related problems may therefore be viewed as establishing a new connection between a fundamental conjecture in complexity theory, and the approximability of a ubiquitous problem in artificial intelligence.

1.2 Width Parameters of Graphs

As mentioned earlier, determining the exact Treewidth of a graph and producing an associated optimal tree decomposition (see Definition 2.1) is known to be NP-hard (Arnborg et al., 1987), and a central open problem is to determine whether or not there exists a polynomial time constant factor approximation algorithm for Treewidth (see e.g., Bodlaender et al., 1995; Feige et al., 2005; Bodlaender, 2005). The current best polynomial time approximation algortihm for Treewidth (Feige et al., 2005), computes the Treewidth $\mathsf{tw}(G)$ within a factor $O(\sqrt{\log \mathsf{tw}(G)})$. On the other hand, the only hardness result to date for Treewidth shows that it is NP-hard to compute Treewidth within an *additive* error of n^{ϵ} for some $\epsilon > 0$ (Bodlaender et al., 1995). No hardness of approximation is known and not even the possibility of a polynomial-time approximation scheme for Treewidth has been ruled out. In many important special classes of graphs, such as planar graphs (Seymour & Thomas, 1994), asteroidal triple-free graphs (Bouchitté & Todinca, 2003), and H-minor-free graphs (Feige et al., 2005), constant factor approximations are known, but the general case has remained elusive.

On the positive side, there is a large body of literature developing fixed-parameter algorithms for Treewidth. Exactly determining the Treewidth is fixed-parameter tractable: there is a linear time algorithm for computing the (exact) Treewidth for graphs of constant treewidth (Bodlaender, 1996). More specifically this exact algorithm runs in time $2^{poly(k)}poly(n)$. Constant factor approximation algorithms achieve better dependence on the treewidth, k, and n, with the best such algorithm running in time $2^{O(k)}O(n)$ (Bodlaender, 2007).

A related graph parameter is the so-called *Pathwidth*, which can be viewed as measuring how close G is to a path. The Pathwidth $\mathsf{pw}(G)$ is always at least $\mathsf{tw}(G)$, but can be much larger. The current state of affairs here is similar as for Treewidth; though the current best approximation algorithm only has an approximation ratio of $O(\sqrt{\log \mathsf{pw}(G)}\log n)$ (Feige et al., 2005), the best hardness result is NP-hardness of additive n^{ϵ} error approximation.

Using the recently proposed *Small Set Expansion* (SSE) Conjecture (Raghavendra & Steurer, 2010) discussed earlier, we show that both $\mathsf{tw}(G)$ and $\mathsf{pw}(G)$ are hard to approximate within any constant factor. In fact, we show something stronger: it is hard to distinguish graphs with small Pathwidth from graphs with large Treewidth. Specifically:

Theorem 1.1. *For every $\alpha > 1$ there is a $c > 0$ such that given a graph $G = (V, E)$ it is SSE-hard to distinguish between the case when $\mathsf{pw}(G) \leq c \cdot |V|$ and the case when $\mathsf{tw}(G) \geq \alpha \cdot c \cdot |V|$.*

In particular, both Treewidth and Pathwidth are SSE-hard to approximate within any constant factor.

This is the first result giving hardness of (relative) approximation for these problems, and gives evidence that no constant factor approximation algorithm exists for either of them.

1.3 Minimum Fill-In

A closely related graph theoretic property is the Minimum Fill-In of a graph, the minimum number of edges required to add to a graph to triangulate it (i.e., make it chordal).

This property has important applications with sparse matrix computations (and in particular Gaussian elimination) and artificial intelligence (see the excellent survey in Heggernes, 2006).

Minimum Fill-in has been known to be fixed parameter tractable since 1994, when Kaplan et al. (1994) gave an $O(|E|16^k)$ algorithm, where k is the number of edges required. From there, several improvements to the running time have been given, with the most recent in 2012 by Fomin and Villanger (2012), who gave the first subexponential parameterized algorithm, running in time $O(2^{O(\sqrt{k}\log k)} + k^2|V| \cdot |E|)$. In the work of Natanzon et al. (1998), a polynomial time approximation algorithm was presented, which computed a value at most $8k^2$, where k is the optimal solution. For graphs with degree bounded by d, their algorithm achieves an approximation ratio of $O(d^{2.5}\log^4(kd))$.

This remains the best polynomial time approximation algorithm known to date. In particular, it has remained an open question whether a polynomial time constant factor approximation algorithm exists. In this paper, we show that this is not possible, assuming the SSE Conjecture.

Theorem 1.2. *It is SSE-hard to approximate the Minimum Fill-In of a graph to within a constant factor.*

1.4 Pebbling Problems

Graph pebbling is a rich and relatively mature topic in theoretical computer science. *Pebbling* is a game defined on a directed acyclic graph (DAG), where the goal is to *pebble* the sink nodes of the DAG according to certain rules, using the minimum number of pebbles. The rules for pebbling are as follows. A *black pebble* can be placed on a node if all of the node's immediate predecessors contain pebbles, and can always be removed. A white pebble can always be placed on a node, but can only be removed if all of the node's immediate predecessors contain pebbles. A pebbling *strategy* is a process of pebbling the sink nodes in a graph according to the above rules. The *pebbling cost* of a pebbling strategy is the maximum number of pebbles used in the strategy. The Black-White pebbling cost of a DAG is the minimum pebbling cost of all possible pebbling strategies. The black pebbling cost is the minimum pebbling cost over all pebbling strategies that only use black pebbles.

Pebbling games were originally devised for studying programming languages and compiler construction, but have later found a broad range of applications in computational complexity theory. Pebbling is a tool for studying the relationship between computation time and space by means of a game played on directed acyclic graphs. It was employed to model register allocation, and to analyze the relative power of time and space as Turing machine resources. For a comprehensive recent survey on graph pebbling, see the work of Nordström (2010).

Apart from the cost of a pebbling, another important measure is the *pebbling time*, which is the number of steps (pebble placements/removals) performed. In the context of measuring memory used by computations, this corresponds to computation time, and hence keeping the pebbling time small is a natural priority. The extreme case of this is what we refer to as *One-Shot Pebbling*, also known as progressive pebbling, considered in the literature (e.g., Sethi, 1973; Lengauer, 1981; Kirousis & Papadimitriou, 1986). In One-Shot Pebbling, we have the restriction that each node can receive a pebble only once. Note that this restriction can cause a huge increase in the pebbling cost of the graph (Lengauer & Tarjan, 1982).

The One-Shot Pebbling problem is easier to analyze for the following reasons. In the original pebbling problem, in order to achieve the minimum pebbling number, the pebbling time might be required to be exponentially long, which becomes impractical when n is large. On the other hand, the One-Shot Pebbling problem is more amenable to complexity theoretic analysis as it minimizes the space used in a computation subject to the execution time being minimum. In particular, the decision problem for One-Shot Pebbling is in NP (whereas the unrestricted pebbling problems are PSPACE-complete).

The One-Shot Black Pebbling problem and One-Shot Black-White Pebbling problems admit an $O(\sqrt{\log n}\log n)$ approximation ratio. We show that they are SSE-hard to approximate to within any constant factor. For black pebbling we show that this holds for single sink DAGs with in-degree 2, which is the canonical setting for pebbling games (it seems plausible that the black-white hardness can be shown to hold for this case as well, though we have not attempted to prove this).

Theorem 1.3. *It is SSE-hard to approximate the One-Shot Black Pebbling problem within any constant factor, even in DAGs with a single sink and maximum in-degree* 2.

Theorem 1.4. *It is SSE-hard to approximate the One-Shot Black-White Pebbling problem within any constant factor.*

No hardness of approximation result of any form was known for One-Shot Pebbling problems. We believe that these results can be extended to obtain hardness for more relaxed versions of bounded time pebbling costs as well. We are currently working on this, and have some preliminary results.

1.5 The Connection: Layout Problems

The graph width and One-Shot Pebbling problems discussed in the previous sections may at first glance appear to be unrelated. However, both sets of problems are instances of a general family of problems, known as *graph layout problems*. In a graph layout problem (also known as an arrangement problem, or a vertex ordering problem), the goal is to find an ordering of the vertices, optimizing some condition on the edges, such as adjacent pairs being close. Layout problems are an important class of problems that have applications in many areas such as VLSI circuit design.

A classic example is the *Minimum Cut Linear Arrangement* problem (MCLA). In this problem, the objective is to find a permutation π of the vertices V of an undirected graph $G = (V, E)$, such that the largest number of edges crossing any point,

$$\max_i |\{(u, v) \in E|\pi(u) \le i < \pi(v)\}|, \tag{1}$$

is minimized. MCLA is closely related to the *Minimum Linear Arrangement* problem (MLA), in which the max in (1) is replaced by a sum.

The MCLA problem can be approximated to within a factor $O(\log n\sqrt{\log n})$. To the best of our knowledge, there is no hardness of approximation for MCLA in the literature. Its cousin MLA was recently proved SSE-hard to approximate within any constant factor

(Raghavendra et al., 2012), and we observe that the same hardness applies to the MCLA problem.

Theorem 1.5. *It is SSE-hard to approximate the Minimum Cut Linear Arrangement problem within any constant factor.*

Another example of graph layout is the *Interval Graph Completion* Problem (IGC). In this problem, the objective is to find a supergraph $G' = (V, E')$ of G with the same vertex set V, such that G' is an interval graph (i.e., the intersection graph of a set of intervals on the real line) and having minimum number of edges. While not immediately appearing to be a layout problem, using a simple structural characterization of interval graphs (Ramalingam & Rangan, 1988) one can show that IGC can be reformulated as finding a permutation of the vertices that minimizes the sum over the longest edges going out from each vertex, i.e., minimizing

$$\sum_{u \in V} \max_{(u,v) \in E} \max\{\pi(v) - \pi(u), 0\}. \tag{2}$$

See, for example, the work of Charikar et al. (2010). The current best approximation algorithm for IGC achieves a ratio of $O(\sqrt{\log n} \log \log n)$ (Charikar et al., 2010). It turns out that the SSE Conjecture can be used to prove super-constant hardness for this problem as well.

Theorem 1.6. *It is SSE-hard to approximate the Interval Graph Completion problem within any constant factor.*

There is a distinction in IGC of whether one counts the number of edges in the final interval graph – this is the most common definition – or whether one only counts the number of edges added to make G an interval graph (which makes the problem harder from an approximability viewpoint). Our result holds for the common definition and therefore applies also to the harder version. Note that Interval Graph Completion is well connected to Pathwidth: the pathwidth of a graph G is one less than the smallest clique number of an interval graph that contains G as a subgraph.

Theorems 1.5 and 1.6 are just two examples of layout problems that we prove hardness of approximation for. By varying the precise objective function and also considering directed acyclic graphs, in which case the permutation π must be a topological ordering of the graph, one can obtain a wide variety of graph layout problems. We consider a set of eight such problems, generated by three natural variations (see Section 2.3 for precise details), and show super-constant SSE-based hardness for all of them in a unified way. This set of problems includes MLA, MCLA, and IGC, but not problems such as Bandwidth (but on the other hand, strong NP-hardness inapproximability results for Bandwidth are already known Dubey, Feige, & Unger, 2011). See Table 1 in Section 2.3 for a complete list of problems covered.

Theorem 1.7. *Assuming the SSE Conjecture, all problems listed in Table 1 (see page 581) are NP-hard to approximate to within any constant factor.*

Let us now return to the problems discussed in the previous sections. It should not be surprising that the One-Shot Black Pebbling problem is equivalent to a graph layout problem: the one-shot constraint reduces the problem to determining in which order to pebble the vertices; such an ordering induces a pebbling strategy in an obvious way. (Given a graph layout ordering, the vertices can be black pebbled in that order, with each black pebble removed as soon as that vertex is no longer be needed. Conversely, any black pebbling sequence induces the corresponding ordering on the vertices.) For the black-white case, it is known that the One-Shot Black-White Pebbling cost of D is interreducible with a layout problem on an undirected graph G. Both of these layout problems are included in the set of problems we show hardness for, so Theorems 1.3 and 1.4 follow immediately from Theorem 1.7.

Turning to the width parameters, Treewidth is equivalent to a graph layout problem called elimination width. Here the objective function is somewhat more intricate than in the set of basic layout problems we consider in Theorem 1.7, but we are able to extend those results to hold also for elimination width. Pathwidth is also known to be equivalent to a certain graph layout problem, and in fact is equivalent to the layout problem which One-Shot Black-White Pebbling reduces to. We use these connections to prove the hardness of approximation for both Treewidth and Pathwidth, thereby obtaining Theorem 1.1.

1.6 Previous Work

As the reader may have noticed, for all the problems mentioned, the best current algorithms achieve similar poly-logarithmic approximation ratios. Given their close relation, this is of course not surprising. Most of the algorithms are obtained by recursively applying some algorithm for the c-balanced separator problem, in which the objective is to find a bipartition of the vertices of a graph such that both sides contain at least a c fraction of vertices, and the number of edges crossing the partition is minimized.

In the pioneering work on separators by Leighton and Rao (1999), an $O(\log n)$ approximation algorithm for c-balanced separator was given, which was used to design $O(\log^2 n)$ approximation algorithm for a number of graph layout problems such as MLA, MCLA, and Register Sufficiency. Later, Rao and Richa (1998) improved the approximation algorithm for MLA to a ratio $O(\log n \log\log n)$, using a spreading metric method. In the groundbreaking work of Arora et al. (2009), semidefinite programming was used to give an improved approximation ratio of $O(\sqrt{\log n})$ for c-balanced separator. Using their ideas, improved algorithms for ordering problems have been found, such as the $O(\sqrt{\log n}\log\log n)$ approximation algorithm for IGC and MLA (Charikar et al., 2010), the $O(\sqrt{\log n})$ approximation algorithm for Treewidth (Feige et al., 2005) and the $O(\sqrt{\log n}\log n)$ approximation algorithm for Pathwidth (Feige et al., 2005).

It is known that the Register Sufficiency problem (also known as One-Shot Black Pebbling) admits a $O(\log^2 n)$ approximation algorithm (Ravi, Agrawal, & Klein, 1991). We observe that by plugging in the improved approximation algorithm for direct vertex separator (Agarwal, Charikar, Makarychev, & Makarychev, 2005) into the algorithm by Ravi et al. (1991), one can improve this to an $O(\sqrt{\log n}\log n)$ approximation algorithm.

Again, in these algorithms, the approximation algorithm for c-balanced separator plays a key role. An improved algorithm for c-balanced separator will also improve the approximation algorithms for the other problems. On the other hand, hardness of approximating

c-balanced separator (Raghavendra et al., 2012) does not necessarily imply hardness of approximating layout problems.

On the hardness side, our work builds upon the work by Raghavendra et al. (2012), which showed that the SSE Conjecture implies superconstant hardness of approximation for MLA (and for c-balanced separator). The only other hardness of relative approximation that we are aware of for these problems is a result of the work of Ambühl et al. (2007), showing that MLA does not have a PTAS unless NP has randomized subexponential time algorithms.

1.7 Organization

The outline for the rest of the paper is as follows. In Section 2, we formally define the layout problems studied as well as Treewidth, Pathwidth, and Minimum Fill-In. After giving an overview of the reductions used in Section 3 we give the full proof of Theorem 1.7 in Section 4. Then, in Section 5 we give the lower bound on Treewidth which combined with the results from Section 4 gives Theorem 1.1. In Section 6 we give the lower bound on Minimum Fill-In. Finally in Section 7 we give some additional reductions for our pebbling instances in order to achieve indegree 2 and single sinks, as promised in Theorem 1.3. We end with some concluding remarks in Section 8.

2. Definitions and Preliminaries

For an undirected graph $G = (V, E)$, and subsets $S, S' \subseteq V$, $E(S, S')$ denotes the set of edges that go between S and S'. In other words, $E(S, S')$ is the set of edges $(u, v) \in E$ such that $u \in S$ and $v \in S'$.

2.1 Treewidth, Elimination Width, and Pathwidth

Definition 2.1 (Tree decomposition, Treewidth)**.** Let $G = (V, E)$ be a graph, T a tree, and let $\mathcal{V} = (V_t)_{t \in T}$ be a family of vertex sets $V_t \subseteq V$ indexed by the vertices t of T. The pair $(T, \mathcal{V})$ is called a *tree decomposition* of G if it satisfies the following three conditions:

(T1) $V = \cup_{t \in T} V_t$;

(T2) for every edge $e \in E$, there exists a $t \in T$ such that both endpoints of e lie in V_t;

(T3) for every vertex $v \in V$, $\{t \in T \mid v \in V_t\}$ is a subtree of T'.

The width of $(T, \mathcal{V})$ is the number $\max\{|V_t| - 1 \mid t \in T\}$, and the *Treewidth* of G, denoted $\mathsf{tw}(G)$, is the minimum width of any tree decomposition of G.

Definition 2.2. Let $G = (V, E)$ be a graph, and let $v_1, \ldots, v_n$ be some ordering of its vertices. Consider the following process: for each vertex v_i in order, add edges to turn the neighborhood of v_i into a clique, and then remove v_i from G. This is an *elimination ordering* of G. The *width* of an elimination ordering is the maximum over all v_i of the degree of v_i when v_i is eliminated. The *elimination width* of G is the minimum width of any elimination order.

Theorem 2.3 (See e.g., Bodlaender, 2007)**.** *For every graph G, the elimination width of G equals* $\mathsf{tw}(G)$*.*

Thus Treewidth is another example of a layout problem. In principle this layout problem can be formulated in the framework of Section 2.3, but the choice of cost function is now more involved than the vertex- and edge-counting considered there.

Definition 2.4 (Path decomposition, Pathwidth)**.** Given a graph G, we say that $(T, \mathcal{V})$ is a *path decomposition* of G if it is a tree decomposition of G and T is a path. The *Pathwidth* of G, denoted $\mathsf{pw}(G)$, is the minimum width of any path decomposition of G.

As claimed earlier, Pathwidth is in fact equivalent with a graph layout problem. (See the next section for the formal definition of layout.)

Theorem 2.5 (Kinnersley, 1992)**.** *For every graph G, we have* $\mathsf{pw}(G) = \mathsf{Layout}(G; V, \max)$*.*

2.2 Minimum Fill-In

Definition 2.6 (Chordal, Triangulation)**.** A graph G is *chordal* if and only if every cycle of length at least 4 has a chord. For any (possibly non-chordal) graph G, a *triangulation* of G is a supergraph of G which is chordal.

Definition 2.7 (Minimum Fill-In)**.** The *Minimum Fill-In* of a graph G is the minimum number of edges required to add to G to triangulate it; i.e., so that the resulting supergraph is chordal.

The problem of determining the Minimum Fill-In of a graph is sometimes called the Chordal Graph Completion problem.

A *perfect elimination ordering* of G is an elimination ordering such that no edges are ever added to G. Put another way, for each vertex v_i, its neighbours appearing after it in the ordering form a clique.

Theorem 2.8 (Fulkerson & Gross, 1965)**.** *A graph G is chordal if and only if it has a perfect elimination ordering.*

Treewidth and Minimum Fill-In are related through the following theorem.

Theorem 2.9 (Folklore)**.** *Suppose G is a graph with Treewidth k. Then every triangulation of G has a clique of size $k + 1$.*

2.3 Graph Layout Problems

In this subsection, we describe the set of graph layout problems that we consider. A problem from the set is described by three parameters, giving rise to several different problems. These three parameters are by no means the only interesting graph layout problems (and some of the settings give rise to more or less uninteresting layout problems). However, they are sufficient to capture the problems we are interested in except Treewidth, which in principle could be incorporated as well though we refrain from doing so in order to keep the definitions simple (see Section 2.1 for more details).

First a word on notation. Throughout the paper, $G = (V, E)$ denotes an undirected graph, and $D = (V, E)$ denotes a directed (acyclic) graph. Letting n denote the number of vertices of the graph, we are interested in bijective mappings $\pi : V \to [n]$. We say that an edge $(u, v) \in E$ *crosses* point $i \in [n]$ (with respect to the permutation π, which will always be clear from context), if $\pi(u) \le i < \pi(v)$.

We consider the following variations:

1. **Undirected or directed acyclic**: In the case of an undirected graph G, any ordering π of the vertices is a feasible solution. In the case of a DAG D, only the topological orderings of D are feasible solutions.

2. **Counting edges or vertices**: for a point $i \in [n]$ of the ordering, we are interested in the set $E_i(\pi)$ of edges crossing this point. When counting edges, we use the cardinality of E_i as our basic measure. When counting vertices, we only count the set of vertices V_i to the left of i that are incident upon some edge crossing i. In other words, V_i is the projection of $E_i(\pi)$ to the left-hand side vertices. Formally:

$$\begin{aligned} E_i(\pi) &= \{e \in E \mid \pi(u) \le i < \pi(v) \text{ where } e = (u, v)\} \\ V_i(\pi) &= \{u \in V \mid \pi(u) \le i < \pi(v) \text{ for some } (u, v) \in E\} \end{aligned}$$

   We refer to $|E_i(\pi)|$ or $|V_i(\pi)|$ (depending on whether we are counting edges or vertices) as the *cost* of π at i.

3. **Aggregation by sum or max**: given an ordering π, we aggregate the costs of each point $i \in [n]$, by either summation or by taking the maximum cost.

Given these choices, the objective is to find a feasible ordering π that minimizes the aggregated cost.

Definition 2.10. (Layout value) For a graph H (either an undirected graph G or a DAG D), a cost function C (either E or V), and an aggregation function $\text{agg} : \mathbb{R}^* \to \mathbb{R}$ (either Σ or max), we define $\mathsf{Layout}(H; C, \text{agg})$ as the minimum aggregated cost over all feasible orderings of H. Formally:

$$\mathsf{Layout}(H; C, \text{agg}) = \min_{\text{feasible } \pi} \operatorname*{agg}_{i \in [n]} |C_i(\pi)|.$$

Example 2.11.

$$\mathsf{Layout}(G; E, \max) = \min_{\pi} \max_{i \in [n]} |E_i(\pi)|,$$

where π ranges over all orderings of $V(G)$. This we recognize from Section 1.5 as the Minimum Cut Linear Arrangement value of G.

Example 2.12.

$$\mathsf{Layout}(D; V, \max) = \min_{\pi} \max_{i \in [n]} |V_i(\pi)|,$$

where π ranges over all topological orderings of the DAG D. As we shall see in Section 2.4, this is precisely the One-Shot Black Pebbling cost of D.

Problem			Also known as / Equivalent with
undir.	edge	sum	Minimum/Optimal Linear Arrangement
undir.	edge	max	Minimum Cut Linear Arrangement CutWidth
undir.	vertex	sum	Interval Graph Completion SumCut
undir.	vertex	max	Pathwidth One-Shot Black-White Pebbling
DAG	edge	sum	Minimum Storage-Time Sequencing Directed MLA/OLA
DAG	edge	max	
DAG	vertex	sum	
DAG	vertex	max	One-Shot Black Pebbling Register Sufficiency

Table 1: Taxonomy of Layout Problems

Combining the different choices gives rise to a total of eight layout problems (some more natural than others). Several of these appear in the literature under one or more names, and some turn out to be equivalent[1] to problems that at first sight appear to be different. We summarize some of these names in Table 1. In some cases the standard definitions of these problems look somewhat different than the definition given here (e.g., for Pathwidth, One-Shot Pebbling, and Interval Graph Completion). For the Pebbling and Pathwidth problems, we discuss these equivalences of definitions in the following two sections.

For Interval Graph Completion, recall from Section 1.5 that the objective is to minimize

$$\sum_{u \in V} \max_{(u,v) \in E} \max\{\pi(v) - \pi(u), 0\}.$$

In other words, we are counting the longest edge going to the right from each point i. If the length of this edge is l then the edge contributes 1 to $V_i(\pi), \ldots, V_{i+l-1}(\pi)$ and hence the objective can be rewritten as

$$\sum_{u \in V} |V_i(\pi)|,$$

so that Interval Graph Completion is precisely $\mathsf{Layout}(G; V, \Sigma)$.

2.4 Pebbling Problems

In this section we define pebbling problems and their one-shot versions.

Definition 2.13. (Pebbling Configurations) Let $D = (V, E)$ be a directed acyclic graph (DAG). A *pebbling configuration* of D is a pair (B, W) of (disjoint) subsets of vertices (representing the set B of vertices that have black pebbles, and the set W of vertices that have white pebbles on them).

1. Here, we consider two optimization problems equivalent if there are reductions between them that change the objective values by at most an additive constant.

Definition 2.14. (Black and Black-White Pebbling Strategies) Let $D = (V, E)$ be a directed acyclic graph. A *Black-White Pebbling strategy* for D is a sequence of pebble configurations $\mathcal{P} = \{P_0, \ldots, P_\tau\}$ such that:

(i) the first and last configurations contain no pebbles; that is $P_0 = P_\tau = (\emptyset, \emptyset)$.

(ii) each sink vertex u of D is pebbled at least once, i.e., there is some $P_t = (B_t, W_t)$ such that $u \in B_t \cup W_t$.

(iii) each configuration follows from the previous configuration by one of the following rules:

 (a) A black pebble can be removed from a vertex.

 (b) A black pebble can be placed on a pebble-free vertex v if all of the immediate predecessors of v are pebbled.

 (c) A white pebble can be placed on a pebble-free vertex.

 (d) A white pebble can be removed from a vertex v if all of the immediate predecessors of v are pebbled.

A Black Pebbling Strategy for G is a Black-White Pebbling strategy in which no white pebbles are used.

The *cost* of a pebbling strategy is $cost(\mathcal{P}) = \max_{0 \le t \le \tau}\{|B_t \cup W_t|\}$. The Black-White Pebbling cost of D is the minimum cost of any Black-White Pebbling strategy of D, and similarly the Black Pebbling cost of D is the minimum cost of any Black Pebbling Strategy of D.

Definition 2.15. (One-Shot Black and One-Shot Black-White Pebbling) A *One-Shot Black (resp. Black-White) pebbling strategy* is a Black (resp. Black-White) Pebbling strategy in which each node is only pebbled once. The One-Shot Black (resp. Black-White) pebbling cost of D, denoted $\mathsf{BP}^{\mathsf{1s}}(D)$ (resp. $\mathsf{BWP}^{\mathsf{1s}}(D)$) is the minimum cost of any One-Shot Black (resp. Black-White) Pebbling strategy of D.

As mentioned in Table 1, One-Shot Pebbling problems can be formulated as Layout problems.

Lemma 2.16. *For every DAG $D = (V, E)$, we have $\mathsf{BP}^{\mathsf{1s}}(D) = \mathsf{Layout}(D, V, \max)$.*

Proof. Suppose π is the optimal ordering of $\mathsf{Layout}(D, V, \max)$, we pebble the vertices according to π. We remove a pebble from vertex u if and only if all of the successors of u are pebbled. Since π is a topological order of D, this is a valid pebbling strategy. It is easy to verify that after pebbling $\pi(i)$, the number of pebbles on the graph is $|V_i(\pi)|$. Therefore the number of pebbles used in the above strategy is $\mathsf{Layout}(D, V, \max)$. On the other hand, suppose Γ is the optimal pebbling strategy, let σ be the ordering of vertices to receive a pebble in Γ. We consider the number of pebbles on the graph after pebbling the i-th vertex in σ. For any vertex u that has a pebble, if the vertex has a successor that has not yet been pebbled, then the pebble on u cannot be removed, since u cannot be pebbled again. Therefore the number of pebbles on the graph is at least $|V_i(\sigma)|$. Thus $\mathsf{BP}^{\mathsf{1s}}(D) \ge \max_{i \in [n]} |V_i(\sigma)| \ge \mathsf{Layout}(D, V, \max)$. □

For One-Shot Black-White Pebbling, we have the following reductions by Lengauer (1981), showing that One-Shot Black-White Pebbling is equivalent to the undirected Max-Vertex Layout Problem.

Lemma 2.17 (Lengauer, 1981)**.** *For a given DAG $D = (V, E)$, let $G_D = (V, E_D)$ be an undirected graph with $E_D = \{(v, w) \mid (v, w) \in E\} \cup \{(v, w) \mid \exists u, (v, u), (w, u) \in E\}$. Then*

$$\mathsf{BWP}^{\mathsf{1s}}(D) = \mathsf{Layout}(G_D, V, \max) - 1.$$

Lemma 2.18 (Lengauer, 1981)**.** *For an undirected graph $G = (V, E)$, let $D_G = (V \cup E, E_G)$ be a DAG with $E_G = \{(v, e) \mid e \in E, v \in V, v \in e\}$. Then*

$$\mathsf{Layout}(G, V, \max) = \mathsf{BWP}^{\mathsf{1s}}(D_G) + 2.$$

2.5 Small Set Expansion Conjecture

In this section we define the SSE Conjecture. Let $G = (V, E)$ be an undirected d-regular graph. For a set $S \subseteq V$ of vertices, we write $\Phi_G(S)$ for the (normalized) edge expansion of S,

$$\Phi_G(S) = \frac{|E(S, V \setminus S)|}{d|S|}$$

The Small Set Expansion Problem with parameters η and δ, denoted $\mathsf{SSE}(\eta, \delta)$, asks if G has a small set S which does not expand or whether all small sets are highly expanding.

Definition 2.19 ($\mathsf{SSE}(\eta, \delta)$)**.** Given a d-regular graph $G = (V, E)$ [2], $\mathsf{SSE}(\eta, \delta)$ is the problem of distinguishing between the following two cases:

Yes There is an $S \subseteq V$ with $|S| = \delta|V|$ and $\Phi_G(S) \leq \eta$.

No For every $S \subseteq V$ with $|S| = \delta|V|$ it holds that $\Phi_G(S) \geq 1 - \eta$.

This problem was introduced by Raghavendra and Steurer (Raghavendra & Steurer, 2010), who conjectured that the problem is hard.

Conjecture 2.20 (Small Set Expansion Conjecture)**.** *For every $\eta > 0$, there is a $\delta > 0$ such that $\mathsf{SSE}(\eta, \delta)$ is NP-hard.*

As has become common for a conjecture like this (such as the Unique Games Conjecture), we say that a problem is *SSE-hard* if it is as hard to solve as the SSE problem. Formally, a decision problem $\mathcal{P}$ (e.g., a gap version of some optimization problem) is *SSE-hard* if there is some $\eta > 0$ such that for every $\delta > 0$, $\mathsf{SSE}(\eta, \delta)$ polynomially reduces to $\mathcal{P}$.

Subsequently, Raghavendra et al. (2012) showed that the SSE Problem can in turn be reduced to a quantitatively stronger form of itself. To state this stronger version, we need to first define Gaussian noise stability.

Definition 2.21. Let $\rho \in [-1, 1]$. We define $\Gamma_\rho : [0, 1] \to [0, 1]$ by

$$\Gamma_\rho(\mu) = \Pr\left[X \leq \Phi^{-1}(\mu) \wedge Y \leq \Phi^{-1}(\mu)\right]$$

where Φ^{-1} is inverse function of normal distribution, and X and Y are jointly normal random variables with mean 0 and covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$.

2. d is a constant

The only property we shall need of Γ_ρ is the following well-known fact on the asymptotic behaviour for ρ close to 1 and μ bounded away from 0.

Fact 2.22. *(Raghavendra et al., 2012) There is a constant $c > 0$ such that for all sufficiently small ϵ and all $\mu \in [1/10, 1/2]$,*

$$\Gamma_{1-\epsilon}(\mu) \leq \mu(1 - c\sqrt{\epsilon}).$$

We can now state the strong form of the SSE Conjecture.

Conjecture 2.23 (SSE Conjecture, Equivalent Formulation)**.** *For every integer $q > 0$ and $\epsilon, \gamma > 0$, it is NP-hard to distinguish between the following two cases for a given d-regular graph $G = (V, E)$*

Yes *There is a partition of V into q equi-sized sets $S_1, \ldots, S_q$ such that $\Phi_G(S_i) \leq 2\epsilon$ for every $1 \leq i \leq q$.*

No *For every $S \subseteq V$, letting $\mu = |S|/|V|$, it holds that $\Phi_G(S) \geq 1 - (\Gamma_{1-\epsilon/2}(\mu) + \gamma)/\mu$.*

For future reference, let us make two remarks about the strong form of the conjecture.

Remark 2.24. *In the **Yes** case of Conjecture 2.23, the number of edges leaving S_i is at most*

$$|E(S_i, V \setminus S_i)|^3 = \Phi_G(S_i) d|S| \leq 4\epsilon|E|/q.$$

In particular, the total number of edges that are not contained in one of the S_i's is at most

$$\frac{1}{2}\sum_i |E(S_i, V \setminus S_i)| \leq 2\epsilon|E|.$$

Remark 2.25. *Using Fact 2.22 we see that, in the **No** case of Conjecture 2.23, we have*

$$\Phi_G(S) \geq c'\sqrt{\epsilon},$$

for some constant $c' > 0$, provided $\mu \in [1/10, 1/2]$ and setting $\gamma \leq \sqrt{\epsilon}$. In particular, for every $|V|/10 \leq |S| \leq 9|V|/10$, we have $|E(S, V \setminus S)| \geq c\sqrt{\epsilon}|E|$ (switching roles of S and $V \setminus S$ for $|S| > |V|/2$), for some constant $c > 0$ (not the same constant as in Fact 2.22).

3. Brief Overview of Reductions

In this section, we give a very brief overview of the reductions used to prove that the layout problems of Table 1 are SSE-hard to approximate within any constant factor. The full details of these reductions can be found in Section 4.

For the two undirected edge problems (i.e., MLA and MCLA), the hardness follows immediately from the strong form of the SSE Conjecture (Conjecture 2.23) – for the case of MLA this was proved in the work of Raghavendra et al. (2012) and the proof for MCLA is similar. This is our starting point for the remaining problems. Unfortunately, the results do

3. $E(S_1, S_2)$ indicates the number of edges between vertex set S_1 and S_2

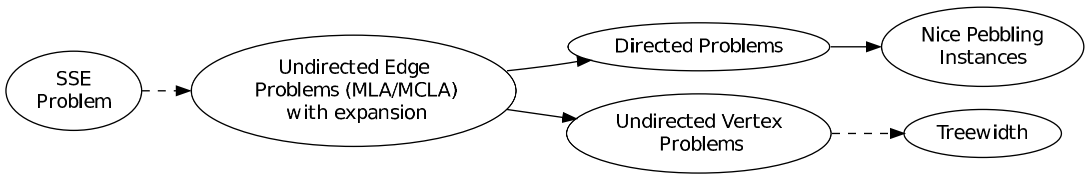

Figure 1: Overview of Reductions. Dashed arrows indicate that the reduction is obtained by the identity mapping, whereas solid arrows indicate a nontrivial transformation from one problem to the other.

not follow from hardness for MLA/MCLA in a black-box way; for the soundness analyses we end up having to use the expansion properties of the original SSE instance.

We then give a reduction from MLA/MCLA to the four directed problems. This reduction simply creates the bipartite graph where the vertex set is the union of the edges and vertices of the original graph G, with directed arcs from an edge e to the vertices incident upon e in G. The use of direction here is crucial: it essentially ensures that both the vertex and edge counts of any feasible ordering corresponds very closely to the number of edges crossing the point in the induced ordering of G.

To obtain hardness for the remaining two undirected problems, we perform a similar reduction as for the directed case, creating the bipartite graph of edge-vertex incidences. However, since we are now creating an undirected graph, we can no longer force the edges to be chosen before the vertices upon which they are incident, which was a key property in the reduction for the directed case. In order to overcome this, we duplicate each original vertex a large number of times. This gives huge penalties to orderings which do not "essentially" obey the desired direction of the edges, and makes the reduction work out.

The results for Treewidth, which are presented in Section 5, follows from an additional analysis of the instances produced by the reduction for undirected vertex problems.

Finally, the reduction for directed problems, implying hardness for One-Shot Black Pebbling, does not produce the kind of "nice" instances promised by Theorem 1.3. In Section 7, we give some additional transformation to achieve these properties.

Figure 1 gives a high-level overview of these reductions.

4. Hardness For Layout Problems

In this section, we show that all of the layout problems defined in Section 2.3 are SSE-hard to approximate within any constant factor. This also shows that Pathwidth and the One-Shot Pebbling problems are hard to approximate within any constant.

4.1 Hardness for MCLA and MLA

In this section, we recall the proof in the work of Raghavendra et al. (2012) for MLA, and observe that it applies to MCLA as well. For an undirected graph G, let us write $\mathsf{MCLA}(G)$

(resp., $\mathsf{MLA}(G)$) for the MCLA value (resp., MLA value) of G, i.e.,

$$\mathsf{MLA}(G) = \mathsf{Layout}(G; E, \Sigma) = \min_{\pi} \sum_{i \in [n]} |E_i(\pi)|$$
$$\mathsf{MCLA}(G) = \mathsf{Layout}(G; E, \max) = \min_{\pi} \max_{i \in [n]} |E_i(\pi)|.$$

Theorem 4.1. *For every $\epsilon > 0$, given a graph $G = (V, E)$, it is SSE-hard to distinguish between:*

Yes $\mathsf{MLA}(G) \leq O(\epsilon \cdot |V| \cdot |E|)$ *and* $\mathsf{MCLA}(G) \leq O(\epsilon |E|)$

No *For every $S \subseteq V$ with $|V|/10 \leq |S| \leq 9|V|/10$, it holds that $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$. In particular,* $\mathsf{MLA}(G) \geq \Omega(\sqrt{\epsilon} \cdot |V| \cdot |E|)$ *and* $\mathsf{MCLA}(G) \geq \Omega(\sqrt{\epsilon}|E|)$.

Proof. We use the instances for Conjecture 2.23 with $q = 1/\epsilon$. Let $G = (V, E)$ be an instance for Conjecture 2.23.

In the **Yes** case, we have disjoint sets $S_1, \ldots, S_q$ and for each set S_j, $|S_j| = n/q = \epsilon n$, $\Phi_G(S_j) \leq 2\epsilon$. We give an ordering $\pi : V \to [1, .., n]$ of the vertices such that $\max_{i \in [n]} |E_i(\pi)| \leq 3\epsilon|E|$ as follows. Order the vertices as $S_1, \ldots, S_q$ (with the order within each S_j chosen arbitrary) and let this order be π. For any $i \in [n]$, we show that $|E_i(\pi)| \leq 3\epsilon|E|$. Suppose $\pi^{-1}(i)$ is a vertex in S_j. Each edge in $E_i(\pi)$ either has both end-points inside S_j, or its end-points in two different S_k's. The total number of edges inside S_j is at most $\epsilon dn/2 = \epsilon|E|$. Moreover, by Remark 2.24, the total number of edges with end-points in two different S_k's is at most $2\epsilon|E|$. Therefore, $\mathsf{MCLA}(G) \leq \max_i |E_i(\pi)| \leq 3\epsilon|E|$. The MLA value can be bounded similarly.

The property of the **No** instance is the same as in Conjecture 2.23 (via Remark 2.25), and the implications for the MLA and MCLA values are immediate. □

4.2 Reduction To Directed Graphs

Given an undirected graph $G = (V, E)$, we construct a directed graph $D = (V', E')$ as follows. In order to distinguish the elements of V and E from the elements of V' and E', we refer to elements of V as vertices, elements of E as edges, elements of V' as nodes, and elements of E' as arcs.

There is a node in D for each vertex and for each edge of G, i.e., $V' = V \cup E$. The graph D is bipartite with bipartition V, E, and there is an arc in D from $e \in E$ to $v \in V$ if e is incident upon v. Formally,

$$\begin{aligned} V' &= V \cup E \\ E' &= \{(e, v) \mid e \in E, v \in V, v \in e\}. \end{aligned}$$

See also Figure 2.

The remainder of this section is devoted to analyzing the reduction. First, it is easy to give an upper bound on the four Layout values of D in terms of the MLA and MCLA values of G.

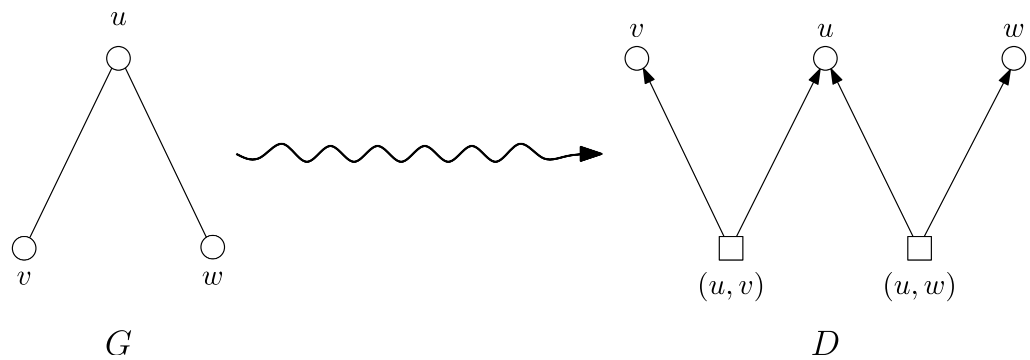

Figure 2: The reduction from G to D.

Lemma 4.2. *The DAG D constructed from G as above satisfies the following:*

$$\mathsf{Layout}(D; E, \Sigma) \leq (\mathsf{MLA}(G) + O(|E|)) \cdot (d+1)$$
$$\mathsf{Layout}(D; E, \max) \leq \mathsf{MCLA}(G) + d.$$

Note that, for the purposes of applying this to the graphs of Theorem 4.1 the error term of $|E|$ (resp. d) is insignifcant compared to the MLA (resp. MCLA) value of G.

Proof. Consider an ordering π of V. For a set of vertices S of V, let $u_\pi(S) \in S$ denote the vertex of S that comes first in the ordering π.

We extend π to an ordering π' of V' by inserting each edge $e = (u, v)$ immediately before the vertex $u_\pi(e)$. It is easy to see that for each node $z \in V'$,

$$|E_z(\pi')| \leq |E_{u_\pi(z)}(\pi)| + d$$

where $E_v(\pi)$ for a vertex v is an abbreviation for $E_{\pi(v)}(\pi)$. This immediately implies

$$\mathsf{Layout}(D; E, \max) \leq \max_{z \in V'} |E_z(\pi')| \leq \max_{u \in V} |E_u(\pi)| + d,$$

Setting π to be an optimal MCLA ordering of G, we obtain the second claim of the Lemma. Similarly, using that $|u_\pi^{-1}(v)| \leq d+1$ for every $v \in V$, we get

$$\mathsf{Layout}(D; E, \Sigma) \leq \sum_{z \in V'} |E_z(\pi')| = \sum_{v \in V} \sum_{\substack{z \in V' \\ u_\pi(z) = v}} |E_z(\pi')| \leq (d+1) \sum_{v \in V} |E_v(\pi)| + d|V'|,$$

Setting π to be an optimal MLA ordering of G and using $|V'| = O(|E|)$, we obtain the first claim of the Lemma. □

Next we use Theorem 4.1 to argue the converse direction.

Lemma 4.3. *Let $0 \leq \epsilon < 1$. Suppose G has the property that for every $|V|/10 \leq |S| \leq 9|V|/10$ we have $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$. Then,*

$$\mathsf{Layout}(D; V, \Sigma) \geq \Omega(\sqrt{\epsilon}|E|^2)$$
$$\mathsf{Layout}(D; V, \max) \geq \Omega(\sqrt{\epsilon}|E|)$$

Proof. Let π' be any ordering of V'. Using the expansion property of Theorem 4.1, we'll show that this ordering must have high cost. For a point $i \in [N]$, let S_i be the set of vertices of V that appear *after* i in π'.

The bound on $\mathsf{Layout}(D; V, \max)$ is immediate: consider a point $i \in [N]$ such that $|S_i| = |V|/2$. By the expansion property $|E(S_i, V \setminus S_i)| \geq \Omega(\sqrt{\epsilon}|E|)$, and since each such edge e has one of its endpoints before or at point i, the node e itself must appear before point i and thus $|V_i(\pi')| \geq |E(S_i, V \setminus S_i)| \geq \Omega(\sqrt{\epsilon}|E|)$.

Let us then turn to $\mathsf{Layout}(D; V, \Sigma)$. Write c_i for the fraction of edges e that appear *before* (or at) point i in π'. We shall show that whenever $1/5 \leq c_i \leq 4/5$, we have $|V_i(\pi')| \geq \Omega(\sqrt{\epsilon}|E|)$, giving a total of $\mathsf{Layout}(D; V, \Sigma) \geq \Omega(\sqrt{\epsilon}|E|^2)$.

By a simple counting argument, we have

$$d|S_i| \geq 2(1 - c_i)|E|,$$

implying $|S_i| \geq (1 - c_i)|V|$ which for $c_i \leq 4/5$ is at least a $1/5$ fraction of vertices. If in addition $|S_i| \leq 9|V|/10$, the argument above gives $|V_i(\pi')| \geq \Omega(\sqrt{\epsilon}|E|)$. The remaining case is that $|S_i| \geq 9|V|/10$. But then S_i is incident upon at least a $9/10$ fraction of edges. This implies that the number of edges incident upon S_i, appearing before i in π', are at least $|E|(c_i - 1/10)$ which for $c_i \geq 1/5$ is $\Omega(\sqrt{\epsilon}|E|)$. □

Combining Lemma 4.2 and Lemma 4.3, with Theorem 4.1, and using the fact that edge costs are always larger than the corresponding vertex costs, we immediately obtain the following theorem.

Theorem 4.4. *Given a DAG D, $\mathsf{Layout}(D; E, \max)$, $\mathsf{Layout}(D; E, \Sigma)$, $\mathsf{Layout}(D; V, \max)$, and $\mathsf{Layout}(D; V, \Sigma)$ are all SSE-hard to approximate within any constant factor, even in DAG's with maximum path length 1 (i.e., every vertex is a source or a sink).*

Proof. Given a graph G, Theorem 4.1 says that it is SSE-hard to distinguish between

Yes $\mathsf{MLA}(G) \leq O(\epsilon \cdot |V| \cdot |E|)$ and $\mathsf{MCLA}(G) \leq O(\epsilon|E|)$

No For every $S \subseteq V$ with $|V|/10 \leq |S| \leq 9|V|/10$, it holds that $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$. In particular, $\mathsf{MLA}(G) \geq \Omega(\sqrt{\epsilon} \cdot |V| \cdot |E|)$ and $\mathsf{MCLA}(G) \geq \Omega(\sqrt{\epsilon}|E|)$.

Applying the reduction to directed graphs, Lemma 4.2 tells us that the **Yes** case becomes

$$\begin{aligned} \mathsf{Layout}(D; E, \Sigma) &\leq (d+1)O(\epsilon|V||E| + |E|) = O(\epsilon|E|^2) \\ \mathsf{Layout}(D; E, \max) &\leq O(\epsilon|E|), \end{aligned}$$

and Lemma 4.3 tells us that the **No** case becomes

$$\begin{aligned} \mathsf{Layout}(D; V, \Sigma) &\geq \Omega(\sqrt{\epsilon}|E|^2) \\ \mathsf{Layout}(D; V, \max) &\geq \Omega(\sqrt{\epsilon}|E|). \end{aligned}$$

thereby establishing a hardness factor of $\Omega(1/\sqrt{\epsilon})$ for each of the four problems (using that $\mathsf{Layout}(D; V, \cdot) \leq \mathsf{Layout}(D; E, \cdot)$). □

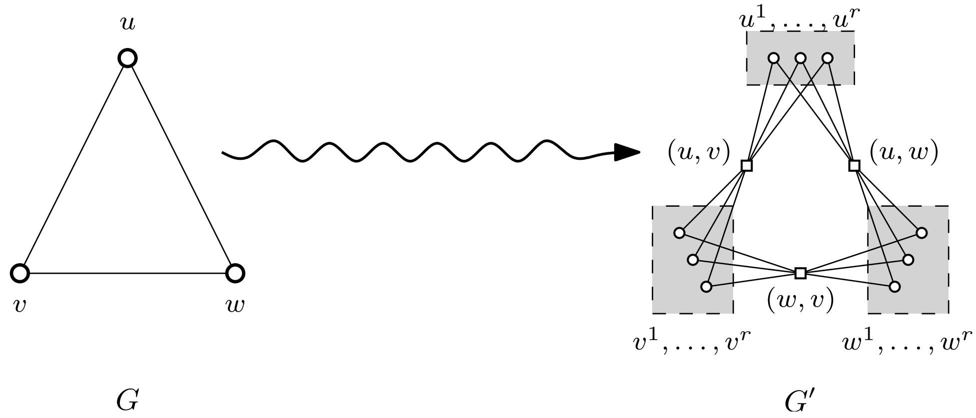

Figure 3: The reduction from G to G', illustrated for $r = 3$.

Remark 4.5. *In fact we see that, as in Theorem 4.1, the four hardness results apply to the same instance, so that it is SSE-hard to distinguish all of the four* Layout *values being high from all of them being low.*

As the One-Shot Black Pebbling problem is precisely $\mathsf{Layout}(D; V, \max)$, we obtain hardness for One-Shot Black Pebbling as an immediate corollary. However, the instances are not single-sink DAGs with maximum indegree 2, as promised in Theorem 1.3. In Section 7 we show how to transform the instances further to obtain such DAGs.

4.3 Undirected Vertex Problems

The reduction for undirected vertex problems is very similar to the reduction for directed problems given in the previous section. As before, we introduce nodes for every edge of G. As in the directed case, we are interested in orderings where an edge appears before its two endpoints, but we cannot use direction to force this anymore. Instead, we ensure that orderings that are not like this incur a high cost by replicating each node corresponding to a vertex of G many times.

Given an undirected graph $G = (V, E)$, we construct a new graph $G' = (V', E')$ as follows.

There are r nodes in G' for each vertex and one node for each edge of G, i.e., $V' = V \times [r] \cup E$. For a vertex $u \in V$ we write $u^1, \ldots, u^r$ to denote the r copies of u and refer to each such set of r nodes as a *vertex group*. The graph G' is bipartite with bipartition $V \times [r], E$, and there is an edge in G' between $e \in E$ and $v^i \in V \times [r]$ if e is incident upon v. Formally,

$$
\begin{aligned}
V' &= \{v^i \mid v \in V, i \in [r]\} \cup \{e \mid e \in E\} \\
E' &= \{(e, v^i) \mid e \in E, v \in V, v \in e, i \in [r]\}.
\end{aligned}
$$

See also Figure 3.

Lemma 4.6. *The graph G' constructed from G as above satisfies the following:*

$$\mathsf{Layout}(G'; V, \Sigma) \leq (d+r)\,\mathsf{MLA}(G)$$
$$\mathsf{Layout}(G'; V, \max) \leq \mathsf{MCLA}(G).$$

Proof. We proceed as in the proof of Lemma 4.2. An ordering π of V naturally induces an ordering π' of V': put all r copies of $u \in V$ consecutively, with vertices of V appearing in the same order as in π, and insert each edge $e \in E$ immediately before its first vertex. Again, for an edge $e \in E$, let $u_\pi(e)$ denote the endpoint of e that appears first in π. Similarly, for a copy $v^i \in V'$ of $v \in V$, let $u_\pi(v^i) = v$. It is easy to see that the constructed ordering π' satisfies

$$|V_z(\pi')| \leq |E_{u_\pi(z)}(\pi)|$$

for every $z \in V'$. This immediately implies

$$\mathsf{Layout}(G'; V, \max) \leq \max_{z \in V'} |V_z(\pi')| \leq \max_{u \in V} |E_u(\pi)|,$$

Similarly, as in Lemma 4.2 we use that $|u_\pi^{-1}(\pi)| \leq d + r$ and get

$$\mathsf{Layout}(G'; V, \Sigma) \leq \sum_{z \in V'} |V_z(\pi')| \leq (d+r) \sum_{v \in V} |E_v(\pi)|.$$

□

Lemma 4.7. *Let $0 \leq \epsilon < 1$. Suppose G has the property that for every $|V|/10 \leq |S| \leq 9|V|/10$ we have $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$. Then, if $r \geq |V| \cdot |E|$, we have*

$$\mathsf{Layout}(G'; V, \Sigma) \geq \Omega(\sqrt{\epsilon} \cdot r \cdot |V| \cdot |E|)$$
$$\mathsf{Layout}(G'; V, \max) \geq \Omega(\sqrt{\epsilon}|E|)$$

Proof. Let π' be an ordering of V'. First we have the following simple claim, establishing that for good orderings, most vertices appear after their edges.

Claim 4.8. *Suppose that for some vertex $u \in V$, at least $r/2$ of the copies of u in G' appear before some edge $e = (u, v) \in E$ adjacent upon u. Then*

$$\max_{i \in [N]} |V_i(\pi')| \geq r/4 \gg \Omega(\sqrt{\epsilon} \cdot |E|)$$
$$\sum_{i \in [N]} |V_i(\pi')| \geq (r/4)^2 \gg \Omega(\sqrt{\epsilon} \cdot r \cdot |V| \cdot |E|).$$

Proof. Let I_1 be the first half of the positions where copies of u appear before e, and I_2 the second half. Thus, $|I_1|, |I_2| \geq r/4$. Then each element of I_1 contributes to $V_i(\pi')$ for each $i \in I_2$, giving the claimed bounds. □

Thus we may without loss of generality assume that for each vertex u of V, at least $r/2$ of its r copies in G' appear *after* all edges adjacent upon u. From now on, let us discard all the $\leq r/2$ "bad" copies of each vertex of V that appear before some of its edges. This only decreases the cost of π', and there are still $\geq r|V|/2$ vertex nodes left.

Let i_1 be the (first) point of π' such that $r|V|/10$ vertex nodes are to the left of i_1, and i_2 the (last) point of π' such that $r|V|/10$ vertex nodes are to the right of i_2.

Claim 4.9. *For any point i between i_1 and i_2, we have $|V_i(\pi')| \geq \Omega(\sqrt{\epsilon}|E|)$.*

Proof. Let $S \subseteq V$ (resp. $T \subseteq V$) be the set of vertices u such that some copy of u appears before i (resp. after i). We then have $|S|, |T| \geq \frac{r|V|/10}{r/2} \geq |V|/5$, and $S \cup T = V$. Thus we can partition V into $S' \subseteq S, T' \subseteq T$ such that $|S'|, |T'| \leq 4|V|/5$. By the expansion property of G we have $|E(S', T')| \geq \Omega(\sqrt{\epsilon}|E|)$. Further, we also have $|V_i(\pi')| \geq |E(S', T')|$ as each $e \in E(S', T')$ must appear before i in π' (because one of their endpoints is in S) but have an edge crossing i (because the other of their endpoints is in T). □

From Claim 4.9, the proof of the lemma follows immediately. □

As in the previous section, we can now combine Lemma 4.6 and Lemma 4.7, with Theorem 4.1, to obtain:

Theorem 4.10. *Given a graph G, $\mathsf{Layout}(G; V, \max)$, $\mathsf{Layout}(G; V, \Sigma)$ are both SSE-hard to approximate within any constant factor, even in bipartite graphs.*

As the Pathwidth problem is precisely $\mathsf{Layout}(G; V, \max)$, we obtain hardness for Pathwidth as an immediate corollary. In the next section, we'll show the stronger soundness required for Theorem 1.1.

5. Hardness For Treewidth

In this section we shall complete our proof of Theorem 1.1 by showing that the hard instances for Pathwidth from Theorem 4.10 also have large Treewidth.

Lemma 5.1. *Let $0 \leq \epsilon < 1$. Let $G = (V, E)$ be an undirected graph with the property that for every $|V|/10 \leq |S| \leq 9|V|/10$ we have $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$, and let G' be the graph obtained by applying the reduction of Section 4.3 to G. Then, if $r \geq |V| \cdot |E|$, we have*

$$\mathsf{tw}(G') \geq \Omega(\sqrt{\epsilon}|E|)$$

To prove Lemma 5.1, we shall use the fact that the Treewidth of a graph is closely related to an expansion-like property called the 1/2-separator number, defined in the work of Bodlaender et al. (1995).

Definition 5.2 (1/2-vertex separator, 1/2-separator number)**.** Let $G = (V, E)$ be an undirected graph. For $W \subseteq V$, a 1/2-*vertex separator* of W in G is a set $S \subseteq V$ of vertices such that every connected component of the graph $G[V - S]$ contains at most $|W|/2$ vertices of W. Let $\psi_G(1/2, W)$ denote the minimum size of a 1/2-vertex separator of W in G. We define the *1/2-separator number* $K_{1/2}(G)$ to be

$$K_{1/2}(G) = \max_{W \subseteq V} \psi_G(1/2, W).$$

Lemma 5.3 (Bodlaender et al., 1995)**.** *For every graph $G = (V, E)$, it holds that* $\mathsf{tw}(G) \geq K_{1/2}(G) - 1$.

Using this, it is now straightforward to prove the lower bound on the Treewidth.

Proof of Lemma 5.1. We'll show that $\psi_{G'}(1/2, V') \geq \Omega(\sqrt{\epsilon}|E|)$ (i.e. we choose $W = V'$). Suppose C is an optimal 1/2-vertex separator of V' and it separates $V' \setminus C$ into l sets $V'_1, \ldots, V'_l$, each of size at most $|V'|/2$. We can merge two sets in $V'_1, \ldots, V'_l$ by combining the vertex set of them. By merging different V'_i we may assume that we only have two sets V'_1 and V'_2, both of size *at least* $|V'|/5$ (this can be achieved by always merging the two sets with smallest number of vertices, whever there are more than two sets left).

Now, similarly to the proof of Claim 4.9, let $S \subseteq V$ (resp. $T \subseteq V$) be the set of vertices v such that some copy of V appears in V'_1 (resp. V'_2). As $|V'_1|, |V'_2| \geq r|V|/5$, this implies that both $|S|, |T|$ are at least $|V|/5$, and furthermore $S \cup T = V$ (since otherwise all r copies of some vertex are in C, implying $|C| \geq r \gg \Omega(\sqrt{\epsilon}|E|)$). We can thus choose a balanced partition S', T' such that $S' \subseteq S$, $T' \subseteq T$, and we have $|E(S', T')| \geq \Omega(\sqrt{\epsilon}|E|)$. But every edge $e = (u, v)$ such that $u \in S'$ and $v \in T'$ must belong to C, since it is connected (in G') to every copy of u and v. □

6. Hardness for Minimum Fill-In

In this section, we use the previous construction and results for Treewidth to prove the inapproximability of Minimum Fill-In. Specifically, we will show that applying the construction to the SSE **No** instances produces graphs with high Minimum Fill-In, while **Yes** instances yields graphs with low Minimum Fill-In.

Lemma 6.1. *Let $0 \leq \epsilon < 1$. Let $G = (V, E)$ be an undirected graph with the property that for every $|V|/10 \leq |S| \leq 9|V|/10$ we have $|E(S, V \setminus S)| \geq \Omega(\sqrt{\epsilon}|E|)$, and let G' be the graph obtained by applying the reduction of Section 4.3 to G. Then, if $r \geq |V| \cdot |E|$, at least $\Omega(\epsilon|E|^2)$ edges must be added to triangulate G'.*

Proof. We use the observation that G' is bipartite. Consider a minimum triangulation of G, which must have a clique of size $\mathsf{tw}(G') + 1$. By Lemma 5.1, one set of of the bipartition of G must have $\Omega(\sqrt{\epsilon}|E|)$ vertices in this clique, and since these vertices are independent in G', $\Omega(\epsilon|E|^2)$ edges must be added. □

Note that this lemma holds independently of the choice of q. Now we prove a good upper bound on **Yes** instances.

Lemma 6.2. *Let $\epsilon > 0$ and $q = 1/\epsilon^2$. Let $G = (V, E)$ be a d-regular graph, and suppose there is a partition of V into q equi-sized sets $S_1, \ldots, S_q$ such that $\Phi_G(S_i) \leq 2\epsilon$ for every $1 \leq i \leq q$.*

Let $G' = (V', E')$ be the graph obtained by applying the reduction of Section 4.3 to G. Then G' has a triangulation with $|E'| + O(\epsilon^2|E|^2)$ edges.

Suppose $G = (V, E)$ is a **Yes** *instance of Conjecture 2.23 with parameters q, ϵ, with $q = 1/\epsilon^2$, and apply the previous construction with $r \geq |V| \cdot |E|$ to get a graph $G' = (V', E')$. Then G' has a triangulation with $|E'| + O(\epsilon^2|E|^2)$ edges.*

Proof. Note that it suffices to find a "good" ordering of the vertices of G' which is close to being a perfect elimination ordering, i.e., that requires few additional edges to turn it into a perfect elimination ordering. For each i, we define the set

$$E_i = \{(u, v) \in E \mid u, v \in S_i\}.$$

Also define the set of "cut" edges $E_c = \{(u, v) \in E \mid u \in S_i, v \in S_j, i \neq j\}$. Clearly, the E_i together with E_c form a partition of E. Further define subsets of E_c by $E_c^i = E(S_i, V \backslash S_i)$, the set of cut edges of S_i. By the definition of the S_i, $\Phi_G(S_i) \leq 2\epsilon$, and hence $|E_c^i| \leq 4\epsilon|E|/q = 4\epsilon^3|E|$. We'll identify the E_i and E_c as subsets of Y in the constructed graph.

Now let π be an ordering of V' such that all $r|V|$ of the vertex copies appear first (in any order), followed by the sets $E_1, \ldots, E_q, E_c$ in that order, where within each set the vertex order is arbitrary. Now consider the following supergraph H of G' obtained by:

- making each set $E_i \cup E_c^i$ a clique
- making E_c a clique/

Claim 1: This only adds $O(\epsilon^2|E|^2)$ edges.

Note that $|E_i \cup E_c^i| \leq d|S_i| = \frac{d|V|}{q} = \epsilon^2 \frac{|E|}{2}$. Thus making $E_i \cup E_c^i$ a clique requires $O(\epsilon^4|E|^2)$ edges. Since there are $q = \frac{1}{\epsilon^2}$ of these cliques in total, this requires $O(\epsilon^2|E|^2)$ edges total. Finally, by Remark 2.24, $|E_c| \leq 2\epsilon|E|$, so making this a clique takes $O(\epsilon^2|E|^2)$ edges as well.

Claim 2: Adding these edges makes π a perfect elimination ordering for H.

Consider a vertex copy $v^k \in X$. Its neighbours in H are the edges which are incident with v; if $v \in S_i$, then these edges must all be in $E_i \cup E_c^i$. So every vertex in X satisfies the perfect elimination property. Now consider $(u, v) \in E_i \subseteq Y$. Its "edge" neighbours are all in E_i or E_c^i. Finally, for every "edge" vertex $(u, v) \in E_c$, its only neighbours that appear after it with respect to π are also in E_c.

Putting these two claims together yields the desired result. □

Combining these two lemmas prove Theorem 1.2.

7. Nicer Pebbling Instances

In this section we show how to transform our hard instances for One-Shot Black Pebbling so as to have in-degree bounded by 2 and single sinks.

Lemma 7.1. *Given a DAG $D = (V, E)$ we can in polynomial time construct a DAG $D' = (V', E')$ such that D' has a single sink and*

$$\mathsf{BP}^{\mathsf{1s}}(D) \leq \mathsf{BP}^{\mathsf{1s}}(D') \leq \mathsf{BP}^{\mathsf{1s}}(D) + s + 1,$$

where s is the number of sinks in D. Furthermore, the maximum in-degree in D' is no larger than the maximum in-degree in D, provided this quantity is at least 2.

Proof. Construct D' by adding a binary tree with s leaves to D, and identifying the leaves of the tree with the sinks of D. That is, D' consists of a binary tree with s leaves on top of a copy of D where the leaves of the binary tree are identified with the sinks of D. The number of vertices of D' is equal to $|D| + |s| - 1$. Note that any binary tree with s leaves will suffice. The properties of D' are easily verified. Since D' is a super-DAG of D, its pebbling cost must be at least $\mathsf{BP}^{\mathsf{1s}}(D)$. Conversely, a valid pebbling of D' can be obtained by using a One-Shot Pebbling of D but without removing pebbles from the sinks, and then pebbling the tree. □

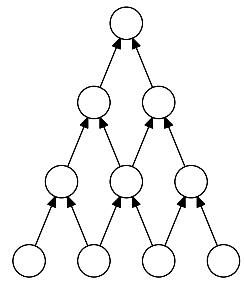

Figure 4: Pyramid of size 4

For the in-degree, we prove the following.

Lemma 7.2. *Given a DAG $D = (V, E)$ we can in polynomial time construct a DAG $D' = (V', E')$ such that every node of D' has in-degree at most 2 and*

$$\mathsf{BP}^{\mathsf{1s}}(D) \leq \mathsf{BP}^{\mathsf{1s}}(D') \leq \mathsf{BP}^{\mathsf{1s}}(D) + d,$$

where d is the maximum in-degree of D. Furthermore, if D has a single sink then so does D'.

Before proving Lemma 7.2, let us see how to use the two Lemmas to derive Theorem 1.3.

Proof of Theorem 1.3. By (the proof of) Theorem 4.4, there is a reduction which takes an instance $(G = (V, E)$ for Conjecture 2.23 and produces a dag D such that $\mathsf{BP}^{\mathsf{1s}}(D) = \mathsf{Layout}(D; V, \max) = O(\epsilon|E|)$ if G is a Yes instance, and $\mathsf{BP}^{\mathsf{1s}}(D) = \Omega(\sqrt{\epsilon}|E|)$ if G is a No instance. The number of sinks in D is $|V| = 2|E|/d$, which is much smaller than $O(\epsilon|E|)$ provided $d \gg 1/\epsilon$ (which may be assumed without loss of generality). The maximum in-degree in D is the maximum degree of G which is $d = O(1)$. Applying the reduction of Lemma 7.1 and then the reduction of Lemma 7.2 we obtain a dag D' with a single sink and in-degree 2 such that $\mathsf{BP}^{\mathsf{1s}}(D) \leq \mathsf{BP}^{\mathsf{1s}}(D') \leq \mathsf{BP}^{\mathsf{1s}}(D) + d + 2|E|/d$. Since $d + 2|E|/d \ll O(\epsilon|E|) \leq \mathsf{BP}^{\mathsf{1s}}(D)$, we conclude that it is SSE-hard to distinguish between $\mathsf{BP}^{\mathsf{1s}}(D') \leq O(\epsilon|E|)$ and $\mathsf{BP}^{\mathsf{1s}}(D') \geq \Omega(\sqrt{\epsilon}|E|)$. $\square$

7.1 Lemma 7.2

In this section we prove Lemma 7.2. First, recall the definition of a pyramid graph.

Definition 7.3. A *pyramid graph* of size d is a layered graph of indegree two, with d layers, labelled $0, 1, \ldots, d-1$. Layer zero (the input layer) consists of d vertices, and layer i contains $d-i$ vertices. Call the vertices at layer i $v_i^1, \ldots v_i^{d-i}$. For all i, $1 \leq i \leq d-1$, $1 \leq j \leq d-i$, Vertex v_i^j has incoming edges from vertices v_{i-1}^j and v_{i-1}^{j+1}. See Figure 4.

The reduction of Lemma 7.2 to produce DAGs of indegree 2 is as follows. Construct D' by replacing each vertex u by a pyramid P_u of size $d(u)$ (here, $d(u)$ denotes the indegree of u), where the $d(u)$ vertices at layer 0 of P_u are identified with the predecessors of u, and u is identified with the vertex at layer $d(u) - 1$ of P_u. See Figure 5.

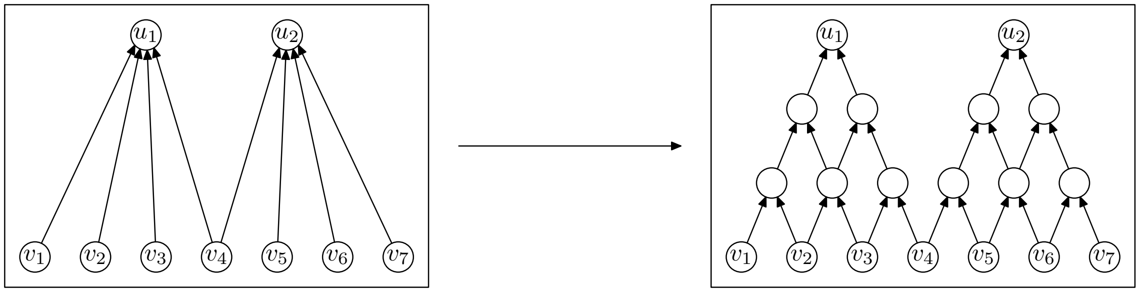

Figure 5: Reduction to DAGs of indegree 2

To prove the lemma we need to show that D' constructed this way satisfies

$$\mathsf{BP}^{1\mathsf{s}}(D) \le \mathsf{BP}^{1\mathsf{s}}(D') \le \mathsf{BP}^{1\mathsf{s}}(D) + d,$$

where d is the maximum indegree of any vertex u of D.

In what follows whenever we say "pebbling strategy" of D or D' we always refer to a One-Shot Black Pebbling strategy of D or D'.

The upper bound on $\mathsf{BP}^{1\mathsf{s}}(D')$ is trivial: if S is a valid pebbling strategy for D, then clearly we can create a corresponding pebbling strategy for D' by pebbling through the pyramid whenever D pebbles the sink of the pyramid. This takes at most d additional pebbles.

In the other direction, we want to show that a pebbling strategy for D' can be converted into a pebbling strategy for D. We first show that D' can be assumed to be in a particular normal form, and then using this normal form, we show how to simulate the pebbling.

Definition 7.4. Let S' be a pebbling strategy of D'. That is, S' is a sequence of configurations, where each configuration is a set of black pebble placements, and such that the sequence of configurations follows the black pebbling rules. We say that configuration $c \in S'$ is saturated with respect to a pyramid P_u if c is the first time in S' that there is a black pebble path cutting the sink of P_u from all of the sources of P_u. (The cut does not include any sources or the sink of P_u.) Note that this cut has size $d-1$.

Claim 7.5. *Let S' be a pebbling strategy for D'. We can assume without loss of generality that S' has the following normal form. For each configuration $c' \in S'$, if c' is saturated with respect to pyramid P_u, then the subsequent moves of S' pebble the sink of P_u (in the obvious way), removing all other black pebbles on the internal nodes of P_u.*

Proof Sketch. We show that any pebbling strategy can be converted into a normal form strategy of the same pebbling cost. At a saturated configuration c', there must be $d-1$ pebbles on internal nodes of P_u. If we subsequently pebble the sink of p, we never use more than $d-1$ pebbles on internal nodes of p, and all other pebbles on the graph stay as they were. Thus the normal form does not use more pebbles than the original strategy. Furthermore, since the internal nodes of a pyramid are only used to pebble the sink of this pyramid, we have not lost anything by pebbling through to the sink and removing the other internal black pebbles. ☐

From now on we assume that the pebbling S' of D' has the above normal form. That is, if a configuration is saturated (with respect to a pyramid P_u), the next thing that happens in S' is to pebble the sink of P_u. (After pebbling the sink, we will have not touched whatever pebbles were on the source nodes of P_u, and we will have a pebble on the sink node of P_u, and no other internal pebbles on P_u.)

Our strategy for constructing a pebbling, S, of D, given a normal form pebbling, S' of D' is as follows. For each node v of D, pebble v whenever it is first pebbled in S', and remove the pebble from v as soon as all successors of v (in the original graph D) are pebbled. We want to argue that the cost of the pebbling strategy S is not greater than that of S'. To see this, we use the following Lemma.

Lemma 7.6. *In any minimal-length One-Shot Black Pebbling of a size d pyramid, the number of pebbles on the pyramid at any point in time, up until all sources are pebbled, must be at least the number of sources in that pyramid that have been pebbled so far.*

Assuming the above Lemma it is clear that if S' is a normal form pebbling of D', then for any pyramid P_u in D', and any configuration c', if there are k pebbles on P_u at c', then in the corresponding configuration c of D, there are at most k pebbles on source nodes of P_u. To see this, first notice that by the above Lemma, any time a pyramid is being pebbled in D' up until the time when all source nodes of the pyramid are pebbled for the first time, the number of pebbles on the pyramid will be at least as large as the number of source nodes in D that contain pebbles. Then by the normal form property of D', as soon as all source nodes of D' are pebbled for the first time, the strategy pebbles the sink of D', and thus again the number of corresponding pebbles on D is never greater than the number of pebbles on D'.

Proof of Lemma 7.6. Let P be a size d pyramid graph, and let S be a One-Shot Black Pebbling of P. Let c be a configuration occurring in S such that the set of source nodes that have been pebbled up to c are the source nodes of P', where P' is a size d' sub-pyramid of P. We want to argue that c must contain at least d' pebbles. Assume without loss of generality that P' is the leftmost sub-pyramid of P, of size $d' < d$ (with source vertices $v_0^1, \ldots, v_0^{d'}$.) Consider the outer rightmost vertices of P' – those vertices with labels $v_{d'-1}^1, v_{d'-2}^2, \ldots, v_1^{d'-1}, v_0^{d'}$. Relabel these outer rightmost vertices of P' by $v_{d-1}, ..., v_0$, where v_{d-1} is the sink vertex of P', and for all $i < d-1$, v_i is the rightmost vertex in P' at level i. Corresponding to each named vertex v_i is a diagonal set of vertices, $diag(v_i)$, beginning at v_i and travelling southwest to a source vertex of P'. Note that the sets $diag(v_i)$ are pairwise disjoint. We will argue that for each i, $0 \leq i \leq d-1$, at least one vertex from $diag(v_i)$ must appear in c. To see this, first notice that for each v_i, there is a vertex v_i' that is an immediate successor of v_i and that lies outside of P'. This vertex v_i' must be pebbled at some point, and furthermore it must be pebbled at some time *after* configuration c, since pebbling v_i' requires pebbling a source vertex of P that is not in P', and the only source vertices that have been pebbled so far are the source vertices of P'. But in order to pebble v_i' in the future, if S is a minimal-length pebbling, then there must be a black pebble on some vertex in $diag(v_i)$ in c. (In a minimal-length pebbling of graph G, if any set of configurations is removed from the pebbling, what results is no longer a black pebbling of G.) Thus, we have shown that if c is any configuration in S such that $d' < d$ source vertices are pebbled thus far, then there must be d' vertices pebbled in c. □

8. Conclusion and Open Problems

We proved SSE-hardness of approximation for a variety of graph problems. Most importantly we obtained the first inapproximability result for the Treewidth problem and Minimum Fill-In.

Some remarks are in order. The status of the SSE conjecture is, at this point in time, very uncertain, and our results should therefore not be taken as absolute evidence that there is no polynomial time approximation algorithm for (e.g.) Treewidth. However, at the very least, our results do give an indication of the difficulty involved in obtaining such an algorithm for Treewidth, and builds a connection between these two important problems. We also find it remarkable how simple our reductions and proofs are. We leave the choice of whether to view this as a healthy sign of strength of the SSE Conjecture, or whether to view it as an indication that the conjecture is too strong, to the reader.

There are many important open questions and natural avenues for further work, including:

1. It seems plausible that these results can be extended to a wider range of graph layout problems. For instance, our two choices of aggregators max and Σ can be viewed as taking ℓ_∞ and ℓ_1 norms, and it seems likely that the results would apply for any ℓ_p norm (though we are not aware of any previous literature studying such variants).

2. It would be nice to obtain hardness of approximation result for our problems based on a weaker hardness assumption such as UGC. It is conjectured in the work of Raghavendra et al. (2012) that the SSE conjecture is equivalent to UGC. Alternatively, it would be nice to show that hardness of some of our problems imply hardness for the SSE Problem.

3. For pebbling, it would be very interesting to obtain results for the unrestricted pebbling problems (for which finding the exact pebbling cost is even PSPACE-hard). As far as we are aware, nothing is known for these problems, not even, say, whether one can obtain a non-trivial approximation in NP. As mentioned in the introduction, we are currently working on extending our One-Shot Pebbling results to bounded time pebblings. We have some preliminary progress there and are hopeful that we can relax the pebbling results to a much larger class of pebblings.

Acknowledgments

Research supported by NSERC.

References

Agarwal, A., Charikar, M., Makarychev, K., & Makarychev, Y. (2005). $O(\sqrt{logn})$ approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM Symposium on Theory of computing*, STOC '05, pp. 573–581, New York, NY, USA. ACM.

Ambuhl, C., Mastrolilli, M., & Svensson, O. (2007). Inapproximability Results for Sparsest Cut, Optimal Linear Arrangement, and Precedence Constrained Scheduling. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '07, pp. 329–337, Washington, DC, USA. IEEE Computer Society.

Amir, E. (2001). Efficient approximation for triangulation of minimum treewidth. In *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, pp. 7–15. Morgan Kaufmann Publishers.

Arnborg, S., Corneil, D. G., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, *8*, 277–284.

Arora, S., Barak, B., & Steurer, D. (2010). Subexponential algorithms for unique games and related problems. In *Annual Symposium on Foundations of Computer Science*, FOCS '10, pp. 563–572.

Arora, S., Lund, C., Motwani, R., Sudan, M., & Szegedy, M. (1998). Proof verification and the hardness of approximation problems. *J. ACM*, *45*(3), 501–555.

Arora, S., Rao, S., & Vazirani, U. (2009). Expander flows, geometric embeddings and graph partitioning. *J. ACM*, *56*, 5:1–5:37.

Arora, S., & Safra, S. (1998). Probabilistic checking of proofs: a new characterization of np. *J. ACM*, *45*(1), 70–122.

Barak, B., Raghavendra, P., & Steurer, D. (2011). Rounding semidefinite programming hierarchies via global correlation. ECCC Report TR11-065.

Bodlaender, H. L., Gilbert, J. R., Hafsteinsson, H., & Kloks, T. (1995). Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, *18*(2), 238–255.

Bodlaender, H. (2007). Treewidth: Structure and algorithms. In Prencipe, G., & Zaks, S. (Eds.), *Structural Information and Communication Complexity*, Vol. 4474 of *Lecture Notes in Computer Science*, pp. 11–25. Springer Berlin / Heidelberg. 10.1007/978-3-540-72951-8_3.

Bodlaender, H. L. (1996). A linear-time algorithm for finding tree-decompositions of Small Treewidth. *SIAM Journal on Computing*, *25*(6), 1305–1317.

Bodlaender, H. L. (2005). Discovering treewidth. In *SOFSEM*, pp. 1–16.

Bouchitté, V., & Todinca, I. (2003). Approximating the treewidth of at-free graphs. *Discrete Applied Mathematics*, *131*(1), 11–37.

Charikar, M., Hajiaghayi, M., Karloff, H., & Rao, S. (2010). l_2^2 spreading metrics for vertex ordering problems. *Algorithmica*, *56*, 577–604.

Dubey, C. K., Feige, U., & Unger, W. (2011). Hardness results for approximating the bandwidth. *J. Comput. Syst. Sci.*, *77*(1), 62–90.

Feige, U., Hajiaghayi, M., & Lee, J. R. (2005). Improved approximation algorithms for minimum-weight vertex separators. In *Proceedings of the thirty-seventh annual ACM Symposium on Theory of computing*, STOC '05, pp. 563–572, New York, NY, USA. ACM.

Fomin, F. V., & Villanger, Y. (2012). Subexponential parameterized algorithm for minimum fill-in. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pp. 1737–1746. SIAM.

Fulkerson, D. R., & Gross, O. A. (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, *15*, 835–855.

Guruswami, V., & Sinop, A. K. (2011). Lasserre hierarchy, higher eigenvalues, and approximation schemes for quadratic integer programming with psd objectives..

Håstad, J. (1999). Clique is hard to approximate withinn 1- ε. *Acta Mathematica*, *182*(1), 105–142.

Håstad, J. (2001). Some optimal inapproximability results. *J. ACM*, *48*(4), 798–859.

Heggernes, P. (2006). Minimal triangulations of graphs: A survey. *Discrete Mathematics*.

Kaplan, H., Shamir, R., & Tarjan, R. E. (1994). Tractability of parameterized completion problems on chordal and interval graphs: Minimum fill-in and physical mapping (extended abstract). *SIAM J. Comput*, *28*, 780–791.

Khot, S., & Regev, O. (2008). Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, *74*(3), 335–349.

Khot, S., & Vishnoi, N. (2005). On the unique games conjecture. In *Annual Symposium on Foundations of Computer Science*, Vol. 46 of *FOCS '05*, p. 3. IEEE COMPUTER SOCIETY PRESS.

Khot, S. (2002). On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing*, STOC '02, pp. 767–775, New York, NY, USA. ACM.

Kinnersley, N. G. (1992). The vertex separation number of a graph equals its path-width. *Information Processing Letters*, *42*(6), 345–350.

Kirousis, L. M., & Papadimitriou, C. H. (1986). Searching and pebbling. *Theor. Comput. Sci.*, *47*, 205–218.

Kloks, T. (1994). *Treewidth: computations and approximations*, Vol. 842. Springer.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques.* MIT press.

Leighton, T., & Rao, S. (1999). Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, *46*, 787–832.

Lengauer, T. (1981). Black-white pebbles and graph separation. *Acta Informatica*, *16*, 465–475. 10.1007/BF00264496.

Lengauer, T., & Tarjan, R. E. (1982). Asymptotically tight bounds on time-space trade-offs in a pebble game. *J. ACM*, *29*, 1087–1130.

Natanzon, A., Shamir, R., & Sharan, R. (1998). A polynomial approximation algorithm for the minimum fill-in problem. In *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing*, STOC '98, pp. 41–47, New York, NY, USA. ACM.

Nordström, J. (2010). New wine into old wineskins: A survey of some pebbling classics with supplemental results. Draft manuscript.

Raghavendra, P. (2008). Optimal algorithms and inapproximability results for every csp?. In *Proceedings of the 40th annual ACM Symposium on Theory of computing*, pp. 245–254. ACM.

Raghavendra, P., & Steurer, D. (2010). Graph expansion and the unique games conjecture. In *Proceedings of the 42nd ACM Symposium on Theory of computing*, STOC '10, pp. 755–764, New York, NY, USA. ACM.

Raghavendra, P., Steurer, D., & Tulsiani, M. (2012). Reductions Between Expansion Problems. IEEE Conference on Computational Complexity.

Ramalingam, G., & Rangan, C. P. (1988). A unified approach to domination problems on interval graphs. *Inf. Process. Lett.*, *27*, 271–274.

Rao, S., & Richa, A. W. (1998). New approximation techniques for some ordering problems. In *Proceedings of the ninth annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pp. 211–218, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Ravi, R., Agrawal, A., & Klein, P. (1991). Ordering problems approximated: single-processor scheduling and interval graph completion. In Albert, J., Monien, B., & Artalejo, M. (Eds.), *Automata, Languages and Programming*, Vol. 510 of *Lecture Notes in Computer Science*, pp. 751–762. Springer Berlin / Heidelberg. 10.1007/3-540-54233-7180.

Reed, B. A. (1992). Finding approximate separators and computing tree width quickly. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Theory of Computing*, STOC '92, pp. 221–228, New York, NY, USA. ACM.

Robertson, N., & Seymour, P. D. (1984). Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, *36*(1), 49–64.

Robertson, N., & Seymour, P. D. (1986). Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, *7*(3), 309–322.

Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence*, *82*(1), 273–302.

Sethi, R. (1973). Complete register allocation problems. In *Proceedings of the fifth annual ACM Symposium on Theory of computing*, STOC '73, pp. 182–195, New York, NY, USA. ACM.

Seymour, P. D., & Thomas, R. (1994). Call routing and the ratcatcher. *Combinatorica*, *14*(2), 217–241.