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

Per Austrin, Toniann Pitassi, and Yu Wu

Department of Computer Science
University of Toronto
{austrin,toni,wuyu}@cs.toronto.edu

Abstract. We study the approximability of a number of graph problems: treewidth and pathwidth of graphs, 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 hard to approximate within any constant factor.

1 Introduction

One of the great accomplishments in the last twenty years in complexity theory has been the development of ideas that has led to a deep understanding of the approximability of an astonishing number of NP-hard optimization problems. More recently, in the last ten years, the formulation of the Unique Games Conjecture (UGC) due to Khot [15] has inspired a remarkable body of work, clarifying the complexity of many optimization problems, and exposing the central role of semidefinite programming in the development of approximation algorithms.

Despite this tremendous progress, for certain expansion problems such as the c-Balanced Separator problem, and graph layout problems such as the Minimum Linear Arrangement (MLA) problem, their approximation status remained unresolved. That is, even assuming the UGC is not known to be sufficient to obtain hardness of approximation for either of these problems. Moreover, the approximability of many other graph layout problems is similarly unresolved, even under the UGC. Intuitively this is because the hard instances for these problems seem to require a certain global structure such as expansion. 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 c-Balanced Separator seems to require a stronger version of UGC, where the instance is guaranteed to have good expansion.

In [21], the Small Set Expansion (SSE) Conjecture was introduced, and it was shown that it implies the UGC, and that the SSE Conjecture follows if

* Research supported by NSERC.

one assumes that the UGC is true for somewhat expanding graphs. In follow-up work by Raghavendra et al. [22], it was shown that the SSE Conjecture is in fact equivalent to the UGC on somewhat expanding graphs, and that the SSE Conjecture implies hardness of approximation for c-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 appear to be beyond the reach of the UGC.

In this paper, we study the approximability of a host of such graph layout problems, including: treewidth and pathwidth of graphs, one-shot black and black-white pebbling, Minimum Cut Linear Arrangement (MCLA) and Interval Graph Completion (IGC). We prove that all of these problems are SSE-hard to approximate to within any constant factor. Our main contributions, giving SSE-hardness of approximation for all of the graph layout problems mentioned above, are described in the following subsections. For all of these problems, no evidence of hardness of approximation was known prior to our results.

It should be noted that the status of the SSE Conjecture is very open at this point. In particular, by the recent result of Arora et al. [3] (see also subsequent work [5, 14]), it has algorithms running in subexponential time. Still, despite this recent progress providing negative evidence against the SSE Conjecture, it remains open, and we think that investigating what open problems in approximability we can show SSE-hardness for is a worthwhile venture.

1.1 Width Parameters of Graphs

The *treewidth* of a graph, introduced by Robertson and Seymour [24, 25], is a fundamental parameter of a graph that measures how close a graph is to being a tree. The concept is very important since problems of small treewidth can usually be solved efficiently by dynamic programming. Indeed, a large body of NP-hard problems (including all problems definable in monadic second-order logic [11]) are solvable in polynomial time and often even linear time on graphs of bounded treewidth. Examples of such optimization problems include finding a maximum independent set or a Hamiltonian cycle in a graph. In machine learning, tree decompositions play a key role in the development of efficient algorithms for fundamental problems such as probabilistic inference, constraint satisfaction and query optimization. (See the excellent survey [8] for motivation, including theoretical as well as practical applications of treewidth.)

The complexity of approximating treewidth is a longstanding open problem. Determining the exact treewidth of a graph and producing an associated optimal tree decomposition (see Definition 2.3) is known to be NP-hard [2]. A central open problem is to determine whether or not there exists a polynomial time constant factor approximation algorithm for treewidth (see e.g., [9, 13, 8]). The current best polynomial time approximation algortihm for treewidth [13], 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 $\epsilon > 0$ [9]. 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 [27] and H-minor-free graphs [13], 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. In particular, when the runtime is allowed to be exponential in the $\mathsf{tw}(G)$ there are constant factor approximations. Furthermore, even exactly determining the treewidth is fixed-parameter tractable: there is a linear time algorithm for computing the (exact) treewidth for graphs of constant treewidth [7].

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 $O(\sqrt{\log \mathsf{pw}(G)} \log n)$ [13], the best hardness result is NP-hardness of additive n^ϵ error approximation.

Using the recently proposed *Small Set Expansion* (SSE) Conjecture [21] 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.

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.

1.2 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 [20].

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 (see e.g., [26, 18, 17]). In one-shot pebbling, we have the restriction that each node can be pebbled only once. Note that this restriction can cause a huge increase in the pebbling cost of the graph [19].

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/black-white pebbling problems admit $O(\sqrt{\log n}\log n)$ approximation ratios. 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 1.2. *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 1.3. *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.

1.3 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* (MCLA) Problem. 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,

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

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 [22], and we observe that the same hardness applies to the MCLA problem.

Theorem 1.4. *Assuming the SSE Conjecture, Minimum Cut Linear Arrangement is hard to approximate 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 such that G' is an interval graph (i.e., the intersection graph of a set of intervals on the real line) and of minimum size. While not immediately appearing to be a layout problem, using a simple structural characterization of interval graphs [23] 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 e.g., [10]. The current best approximation algorithm for IGC achieves a ratio of $O(\sqrt{\log n} \log \log n)$ [10]. It turns out that the SSE Conjecture can be used to prove super-constant hardness for this problem as well.

Theorem 1.5. *Assuming the SSE Conjecture, Interval Graph Completion is hard to approximate 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.

Theorems 1.4 and 1.5 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 2.1 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 [12]). See Table 1 in Section 2.1 for a complete list of problems covered.

Theorem 1.6. *Assuming the SSE Conjecture, all problems listed in Table 1 (see page 20) are 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. 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 1.2 and 1.3 follow immediately from Theorem 1.6.

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 1.6, 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.

1.4 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, An improved algorithm for c-balanced separator will also improve the approximation algorithms for the various layout problems. On the other hand, hardness of approximating c-balanced separator [22] does not necessarily imply hardness of approximating layout problems.

On the hardness side, our work builds upon the work of [22], 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 Ambühl et al. [1], showing that MLA does not have a PTAS unless NP has randomized subexponential time algorithms.

1.5 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 and pathwidth. Section 3 gives a high level overview of the reductions used, and some concluding remarks and open problems are given in Section 4. Full proofs can be found in the full version of the paper [4].

2 Definitions and Preliminaries

2.1 Graph Layout Problems

In this section, 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 2.2 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 $\pi(u) \leq i < \pi(v)$.

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:

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

   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 2.1 *(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* $\mathrm{agg} : \mathbb{R}^* \to \mathbb{R}$ *(either Σ or $\max$), we define* $\mathsf{Layout}(H; C, \mathrm{agg})$ *as the minimum aggregated cost over all feasible orderings of H. Formally:*

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

Example 2.2. $\mathsf{Layout}(G; E, \max) = \min_\pi \max_{i \in [n]} |E_i(\pi)|$, where π ranges over all orderings of $V(G)$. This we recognize from Section 1.3 as the Minimum Cut Linear Arrangement value of G.

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 unified definition given here, e.g., for pathwidth, one-shot pebblings, and interval graph completion).

Table 1. Taxonomy of Layout Problems

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 Vertex Separation
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

2.2 Treewidth and Pathwidth

Definition 2.3 (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:*

[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.

(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*.*

Treewidth can be characterized in terms of elimination width, which is another example of a layout problem (see e.g., [6]). In principle this layout problem can be formulated in the framework of Section 2.1, 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:

Theorem 2.5 ([16]). *For every graph* G, $\mathsf{pw}(G) = \mathsf{Layout}(G; V, \max)$*, also known (among many other names) as the "vertex separation" number of* G*.*

2.3 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 2.6 ($\mathsf{SSE}(\eta, \delta)$). *Given a* d*-regular graph* $G = (V, E)$, $\mathsf{SSE}(\eta, \delta)$ *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 [21], who conjectured that the problem is hard.

Conjecture 2.7 (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 $\mathcal{P}$.

3 Brief Overview of Reductions

We now 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 details of these reductions can be found in the full version of the paper [4].

For the two undirected edge problems (i.e., MLA and MCLA), the hardness follows immediately from the strong form of the SSE Conjecture – for the case of MLA this was proved in [22] and the proof for MCLA is similar. This is our starting point for the remaining problems. Unfortunately, the results do 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 with expansion, 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 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 1.2. We give some additional transformation to achieve these properties in the full version as well.

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

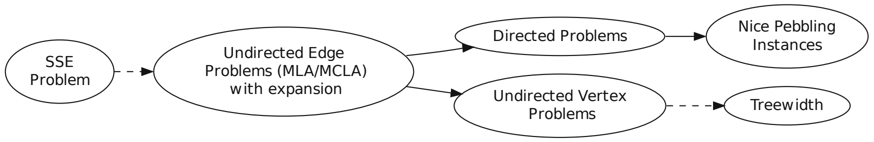
SSE Problem
Undirected Edge Problems (MLA/MCLA) with expansion
Directed Problems
Nice Pebbling Instances
Undirected Vertex Problems
Treewidth

Fig. 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.

4 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. 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 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 [22] 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.

References

[1] Ambuhl, C., Mastrolilli, M., Svensson, O.: Inapproximability Results for Sparsest Cut, Optimal Linear Arrangement, and Precedence Constrained Scheduling. In: Proceedings of the IEEE Symposium on Foundations of Computer Science, pp. 329–337 (2007)
[2] Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM J. Algebraic Discrete Methods 8, 277–284 (1987)
[3] Arora, S., Barak, B., Steurer, D.: Subexponential Algorithms for Unique Games and Related Problems. In: FOCS, pp. 563–572 (2010)
[4] Austrin, P., Pitassi, T., Wu, Y.: Inapproximability of treewidth, one-shot pebbling, and related layout problems. CoRR, abs/1109.4910 (2011)

[5] Barak, B., Raghavendra, P., Steurer, D.: Rounding Semidefinite Programming Hierarchies via Global Correlation. In: FOCS, pp. 472–481 (2011)
[6] Bodlaender, H.L.: Treewidth: Structure and Algorithms. In: Prencipe, G., Zaks, S. (eds.) SIROCCO 2007. LNCS, vol. 4474, pp. 11–25. Springer, Heidelberg (2007)
[7] Bodlaender, H.L.: A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. SIAM J. Comput. 25(6), 1305–1317 (1996)
[8] Bodlaender, H.L.: Discovering Treewidth. In: Vojtáš, P., Bieliková, M., Charron-Bost, B., Sýkora, O. (eds.) SOFSEM 2005. LNCS, vol. 3381, pp. 1–16. Springer, Heidelberg (2005)
[9] Bodlaender, H.L., Gilbert, J.R., Hafsteinsson, H., Kloks, T.: Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. Journal of Algorithms 18(2), 238–255 (1995)
[10] Charikar, M., Hajiaghayi, M., Karloff, H., Rao, S.: l_2^2 spreading metrics for vertex ordering problems. Algorithmica 56, 577–604 (2010)
[11] Courcelle, B.: Graph Rewriting: An Algebraic and Logic Approach. In: Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B), pp. 193–242 (1990)
[12] Dubey, C.K., Feige, U., Unger, W.: Hardness results for approximating the bandwidth. J. Comput. Syst. Sci. 77(1), 62–90 (2011)
[13] Feige, U., Hajiaghayi, M., Lee, J.R.: Improved approximation algorithms for minimum-weight vertex separators. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 563–572 (2005)
[14] Guruswami, V., Sinop, A.K.: Lasserre Hierarchy, Higher Eigenvalues, and Approximation Schemes for Graph Partitioning and Quadratic Integer Programming with PSD Objectives. In: FOCS, pp. 482–491 (2011)
[15] Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the ACM Symposium on Theory of Computing, STOC 2002, pp. 767–775 (2002)
[16] Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. Information Processing Letters 42(6), 345–350 (1992)
[17] Kirousis, L.M., Papadimitriou, C.H.: Searching and pebbling. Theor. Comput. Sci. 47, 205–218 (1986)
[18] Lengauer, T.: Black-white pebbles and graph separation. Acta Informatica 16, 465–475 (1981), doi:10.1007/BF00264496
[19] Lengauer, T., Tarjan, R.E.: Asymptotically tight bounds on time-space trade-offs in a pebble game. J. ACM 29, 1087–1130 (1982)
[20] Nordström, J.: New wine into old wineskins: A survey of some pebbling classics with supplemental results. Draft manuscript (November 2010)
[21] Raghavendra, P., Steurer, D.: Graph expansion and the unique games conjecture. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, pp. 755–764. ACM, New York (2010)
[22] Raghavendra, P., Steurer, D., Tulsiani, M.: Reductions Between Expansion Problems. To appear in CCC (2012)
[23] Ramalingam, G., Rangan, C.P.: A unified approach to domination problems on interval graphs. Inf. Process. Lett. 27, 271–274 (1988)
[24] Robertson, N., Seymour, P.D.: Graph minors. III. Planar tree-width. J. Comb. Theory, Ser. B 36(1), 49–64 (1984)
[25] Robertson, N., Seymour, P.D.: Graph minors. II. Algorithmic aspects of tree-width. Journal of Algorithms 7(3), 309–322 (1986)
[26] Sethi, R.: Complete register allocation problems. In: Proceedings of the Fifth Annual ACM Symposium on Theory of Computing, STOC 1973, pp. 182–195 (1973)
[27] Seymour, P.D., Thomas, R.: Call routing and the ratcatcher. Combinatorica 14(2), 217–241 (1994)