2015 IEEE 56th Annual Symposium on Foundations of Computer Science

Hardness of Approximation in PSPACE and Separation Results for Pebble Games (Extended Abstract)

Siu Man Chan
Department of Computer Science
University of Toronto
Toronto, Canada
siuman@cs.utoronto.ca

Massimo Lauria, Jakob Nordström and Marc Vinyals
School of Computer Science and Communication
KTH Royal Institute of Technology
Stockholm, Sweden
{*lauria,jakobn,vinyals*}*@kth.se*

Abstract

We consider the pebble game on DAGs with bounded fan-in introduced in [Paterson and Hewitt '70] and the reversible version of this game in [Bennett '89], and study the question of how hard it is to decide exactly or approximately the number of pebbles needed for a given DAG in these games.

We prove that the problem of deciding whether s pebbles suffice to reversibly pebble a DAG G is PSPACE-complete, as was previously shown for the standard pebble game in [Gilbert, Lengauer and Tarjan '80]. Via two different graph product constructions we then strengthen these results to establish that both standard and reversible pebbling space are PSPACE-hard to approximate to within any additive constant. To the best of our knowledge, these are the first hardness of approximation results for pebble games in an unrestricted setting (even for polynomial time). Also, since [Chan '13] proved that reversible pebbling is equivalent to the games in [Dymond and Tompa '85] and [Raz and McKenzie '99], our results apply to the Dymond–Tompa and Raz–McKenzie games as well, and from the same paper it follows that resolution depth is PSPACE-hard to determine up to any additive constant.

We also obtain a multiplicative logarithmic separation between reversible and standard pebbling space. This improves on the additive logarithmic separation previously known and could plausibly be tight, although we are not able to prove this.

We leave as an interesting open problem whether our additive hardness of approximation result could be strengthened to a multiplicative bound if the computational resources are decreased from polynomial space to the more common setting of polynomial time.

Keywords

pebbling; reversible pebbling; Dymond-Tompa game; Raz-McKenzie game; PSPACE-complete; separation; PSPACE-hardness of approximation; resolution depth

I. Introduction

In the *pebble game* first studied by Paterson and Hewitt [26], one starts with an empty directed acyclic graph (DAG) G with bounded fan-in (and which in this paper in addition will always have a single sink) and places pebbles on the vertices according to the following rules:

- If all (immediate) predecessors of an empty vertex v contain pebbles, a pebble may be placed on v.
- A pebble may be removed from any vertex at any time.

The goal is to get a pebble on the sink vertex of G with all other vertices being empty, and to do so while minimizing the total number of pebbles on G at any given time (the *pebbling price* of G). This game models computations with execution independent of the actual input. A pebble on a vertex indicates that the corresponding value is currently kept in memory and the objective is to perform the computation with the minimum amount of memory.

0272-5428/15 $31.00 © 2015 IEEE
DOI 10.1109/FOCS.2015.36

The pebble game has been used to study flowcharts and recursive schemata [26], register allocation [33], time and space as Turing-machine resources [10], [18], and algorithmic time-space trade-offs [8], [31], [35]–[37]. In the last 10–15 years, there has been a renewed interest in pebbling in the context of proof complexity as discussed in the survey [24] (although in this context one is often interested also in the slightly more general *black-white pebble game* introduced in [11]), and pebbling has also turned out to be useful for applications in cryptography [1], [12]. An excellent overview of pebbling up to ca. 1980 is given in [28] and some more recent developments are covered in the upcoming survey [25].

Bennett [3] introduced the *reversible pebble game* as part of a broader program [2] to investigate possibilities to eliminate (or significantly reduce) energy dissipation in logical computation. Another reason reversible computation is of interest is that observation-free quantum computation is inherently reversible. In the reversible pebble game, the moves performed in reverse order should also constitute a legal pebbling, which means that the rules for pebble placement and removal become symmetric as follows:

- If all predecessors of an empty vertex v contain pebbles, a pebble may be placed on v.
- If all predecessors of a pebbled vertex v contain pebbles, the pebble on v may be removed.

Reversible pebblings of DAGs have been studied in [19], [22] and have been employed to shed light on time-space trade-offs in reversible simulation of irreversible computation in [4], [20], [21], [39]. In a different line of work Potechin [29] implicitly used the reversible pebble game for proving lower bounds on monotone space complexity, with the connection made explicit in the follow-up works [7], [15].

Another pebble game on DAGs that will be of interest in this paper is the *Dymond–Tompa game* [13] played on a DAG G by a *Pebbler* and a *Challenger*. This game is played in rounds, with both players starting at the sink in the first round. In the following rounds, Pebbler places a pebble on some vertex of G after which Challenger either stays at the current vertex or moves to the newly pebbled vertex. This repeats until at the end of a round Challenger is standing on a vertex with all (immediate) predecessors pebbled (or on a source, in which case the condition vacuously holds), at which point the game ends. Intuitively, Challenger is challenging Pebbler to "catch me if you can" and wants to play for as many rounds as possible, whereas Pebbler wants to "surround" Challenger as quickly as possible. The *Dymond–Tompa price* of G is the smallest number r such that Pebbler can always finish the game in at most r rounds. The Dymond–Tompa game has been used to establish that for parallel time a speed-up by a logarithmic factor is always possible [13], and in [38] it was shown that a slightly modified variant of this game exactly characterizes parallelism in complexity classes like AC^i, NC, and P, and can be used to re-derive, for instance, Savitch's theorem. Furthermore, collapses or separations of these classes can in principle be recast (or discovered) as bounds on Dymond–Tompa price. Interestingly, this characterization of parallelism extends to proof complexity as well as discussed in [5].

A final game with pebbles that we want to just mention without going into any details is the *Raz–McKenzie game* introduced in [30] to study the depth complexity of decision trees solving search problems. The reason for bringing up the Dymond–Tompa and Raz–McKenzie games is that it was shown in [5] that both games are actually equivalent to the reversible pebble game. Hence, any bounds derived for the reversible pebble game also hold for Dymond–Tompa price and Raz–McKenzie price.

The main focus of this paper is to study how hard it is to decide exactly or approximately the pebbling price of a DAG. For the standard pebble game Gilbert et al. [16] showed that given a DAG G and a positive integer s it is PSPACE-complete to determine whether space s is sufficient to pebble G or not. It would seem natural to suspect that reversible pebbling price should be PSPACE-complete as well, but the construction in [16] cannot be used to show this.

Given that pebbling price is hard to determine exactly, an even more interesting question is if anything can be said regarding the hardness of approximating pebbling price. Although this seems like a very natural and appealing question, apparently next to nothing has been known about this.

Wu et al. [40] showed that "one-shot" standard pebbling price is hard to approximate to within any multiplicative constant assuming the so-called Small Set Expansion (SSE) hypothesis. In a one-shot pebbling one is only allowed

to pebble each vertex once, however, and this is a major restriction since the complexity now drops from PSPACE-complete to NP-complete [33]. Note that containment in NP is easy to see since any one-shot pebbling can be described concisely just by listing the order in which the vertices should be pebbled (and it is easy to compute when a pebble is no longer needed and can be removed). In contrast, in the general case pebbling strategies that are optimal with respect to space can sometimes provably require exponential time.

One can also go in the other direction and study more general pebble games, such as the AND/OR pebble game introduced by Lingas [23] in one of the works leading up to [16]. Here every vertex is labelled AND or OR. For AND-vertices we have the usual pebbling rule, but for OR-vertices it is sufficient to just have one pebble on some predecessor in order to be allowed to pebble the vertex. This game has a relatively straightforward reduction from hitting set [14], which shows that it is hard to approximate to within a logarithmic factor, but the reduction crucially depends on the OR-nodes.

We remark that hardness of approximation in PSPACE for other problems has been studied in [9], but those techniques seem hard to adapt to pebble games since the reduction from QBF to pebbling is inherently unable to preserve gaps.

Another problem that we study in the current paper is the relation between standard pebbling price and reversible pebbling price. Clearly, the space needed to reversibly pebble a graph is at least the space required in the standard pebble game. It is also not hard to see that there are graphs that require strictly more pebbles in a reversible setting: for a directed path on n vertices only 2 pebbles are needed in the standard game, while it is relatively straightforward to show that the reversible pebbling space is $\Theta(\log n)$ [3], [22]. However, for "classic" graphs studied in the pebbling literature, such as binary trees, pyramids, certain superconcentrators, and the worst-case graphs in [27], the reversible and standard pebbling prices coincide asymptotically, and are sometimes markedly closer than an additive logarithm apart.

This raises the question whether reversible and standard pebbling can be asymptotically separated with respect to space. It might be worth pointing out in this context that for Turing machines it was proven in [20] that any computation can be simulated reversibly in exactly the same space. In the more restricted pebbling model, it was shown in [19] that if the standard pebbling price of a DAG G on n vertices is s, then G can be reversibly pebbled with at most $s^2 \log n$ pebbles. Thus, if there is not only an additive but also a multiplicative separation between standard and reversible pebbling price, such a separation cannot be too large.

A. Our Results

We obtain the following results:

1. We establish an asymptotic separation between standard and reversible pebbling by exhibiting families of graphs $\{G_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ with a single sink and fan-in 2 which have standard pebbling price $s(n)$ and reversible pebbling price $\Omega(s(n) \log n)$. This construction works for any $s(n) = \mathrm{O}\big(n^{1/2-\epsilon}\big)$ with $\epsilon > 0$ constant, where the constant hidden in the asymptotic notation in the lower bound has a (mild) dependence on ϵ.
2. We prove that determining reversible pebbling price is PSPACE-complete. That is, given a single-sink DAG G of fan-in 2 and a parameter s, it is PSPACE-complete to decide whether G can be reversibly pebbled in space s or not.
3. Finally, we present two different graph products (for standard and reversible pebbling, respectively) that take DAGs G_i of size n_i with pebbling price s_i for $i = 1, 2$ and yield a DAG of size $\mathrm{O}\big((n_1 + n_2)^2\big)$ with pebbling price $s_1 + s_2 + K_p$ (for $K_p = \pm 1$ depending on the flavour of the pebble game). Combining these graph products with the PSPACE-completeness results for standard pebbling in [16] and reversible pebbling in item 2, we deduce that for any fixed K the promise problem of deciding for a DAG G (with a single sink and fan-in 2) whether it can be pebbled in space s or requires space $s + K$ is PSPACE-hard in both the standard and the reversible pebble game.

We need to provide more formal definitions before going into a detailed discussion of techniques, but want to stress right away that a key feature of the above results is the bounded fan-in condition. This is the standard

setting for pebble games in the literature and is also crucial in most of the applications mentioned above. Without this constraint it would be much easier, but also much less interesting, to prove our results.[1]

Another aspect worth pointing out is that although the reversible pebble game is weaker than the standard pebble game, it is technically much more challenging to analyze. The reason for this is that a standard pebbling will always progress in a "forward sweep" through the graph in topological order, and so one can often assume without loss of generality that once one has pebbled through some subgraph the pebbling will never touch this subgraph again. For a reversible pebbling this is not so, since any pebble placed on any descendant of vertices in the subgraph will also have to be removed at some later time, and this has to be done in reverse topological order. Therefore, in any reversible pebbling there will be alternating phases of "forward sweeps" and "reverse sweeps," and these phases can also be interleaved at various levels. For this reason, controlling the progress of a reversible pebbling is substantially more complicated. Despite the additional technical difficulties, however, we consider the reversible pebble game to be at least as interesting to study as the standard and black-white pebble games in view of its tight connection with parallelism in circuit and proof complexity as described in [5].

B. Organization of This Paper

We present the necessary preliminaries in Section II and then give a detailed overview of our results in Section III. Due to space constraints we can only sketch the proofs in this extended abstract and therefore refer to the upcoming full-length version for all missing details. Some concluding remarks are presented in Section IV.

II. Preliminaries

All logarithms in this paper are base 2 unless otherwise specified. For a positive integer n we write $[n]$ to denote the set of integers $\{1, 2, \ldots, n\}$. We use Iverson bracket notation

$$[\![B]\!] = \begin{cases} 1 & \text{if the Boolean expression } B \text{ is true;} \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$

to convert Boolean values to integer values.

A. Boolean Formula Notation and Terminology

A *literal* a over a Boolean variable x is either the variable x itself or its negation $\overline{x}$ (a *positive* or *negative* literal, respectively). A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction of literals. A *k-clause* is a clause that contains at most k literals. A formula F in *conjunctive normal form (CNF)* is a conjunction of clauses $F = C_1 \wedge \cdots \wedge C_m$. A *$k$-CNF formula* is a CNF formula consisting of k-clauses. We think of clauses and CNF formulas as sets, so that the order of elements is irrelevant and there are no repetitions.

A *quantified Boolean formula (QBF)* is a formula $\phi = Q_1 x_1 \, Q_2 x_2 \, \ldots \, Q_n x_n \; F$, where F is a CNF formula over variables $x_1, \ldots, x_n$ and $Q_i \in \{\forall, \exists\}$ are universal or existential quantifiers (i.e., the formula is in prenex normal form with all variables bound by quantifiers). It was shown in [34] that it is PSPACE-complete to decide whether a QBF is true or not (where we can assume without loss of generality that F is a 3-CNF formula).

B. Graph Notation and Terminology

We write $G = (V, E)$ to denote a graph with vertices $V(G) = V$ and edges $E(G) = E$. All graphs in this paper are directed acyclic graphs (DAGs). An edge $(u, v) \in E(G)$ is an *outgoing edge* of u and an *incoming edge* of v, and we say that u is a *predecessor* of v and that v is a *successor* of u. We write $pred_G(v)$ to denote the set of all predecessors of v in G and $succ_G(v)$ to denote all its successors. Vertices with no incoming edges

[1]The reason to emphasize this is that for unbounded fan-in the first author proved a PSPACE-completeness result for reversible pebbling in [6], but this result uses simpler constructions and techniques that do not transfer to the bounded fan-in setting. Another, somewhat related, example is that deciding space in the black-white pebble game has also been shown to be PSPACE-complete for unbounded indegree in [17], but there the unbounded fan-in can be used to eliminate the white pebbles completely, and again the techniques fail to transfer to the bounded indegree case.

are called *sources* and vertices with no outgoing edges are called *sinks*. For brevity, we will sometimes refer to a DAG with a unique sink as a *single-sink DAG*, and this sink will usually be denoted z.

Taking the transitive closures of the predecessor and successor relations, we define the *ancestors* $anc_G(v)$ of v to be the set of vertices that have a path to v and the *descendants* $desc_G(v)$ to be the set of vertices on some path from v. By convention, v is an ancestor and descendant of itself. We write $anc^*_G(v) = anc_G(v) \setminus \{v\}$ and $desc^*_G(v) = desc_G(v) \setminus \{v\}$ to denote the *proper ancestors* and *proper descendants* of v, respectively. These concepts are extended to sets of pairwise incomparable vertices by taking unions so that $anc_G(U) = \bigcup_{u \in U} anc_G(u)$, $anc^*_G(U) = \bigcup_{u \in U} anc^*_G(u)$, et cetera, where we say that the vertices in U are pairwise incomparable when no vertex in the set is an ancestor of any other vertex in the set. When the graph G is clear from context we will sometimes drop it from the notation.

C. Standard and Reversible Pebble Games

A *pebble configuration* on a DAG $G = (V, E)$ is a subset of vertices $\mathbb{P} \subseteq V$. We consider the following three rules for manipulating pebble configurations:

1. $\mathbb{P}' = \mathbb{P} \cup \{v\}$ for $v \notin \mathbb{P}$ such that $pred_G(v) \subseteq \mathbb{P}$ (a *pebble placement* on v).
2. $\mathbb{P}' = \mathbb{P} \setminus \{v\}$ for $v \in \mathbb{P}$ (a *pebble removal* from v).
3. $\mathbb{P}' = \mathbb{P} \setminus \{v\}$ for $v \in \mathbb{P}$ such that $pred_G(v) \subseteq \mathbb{P}$ (a *reversible pebble removal* from v).

A *standard pebbling* from $\mathbb{P}_0$ to $\mathbb{P}_\tau$ is a sequence of pebble configurations $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_\tau)$ where each configuration is obtained from the preceding one by the rules 1 and 2 while in a *reversible pebbling* rules 1 and 3 should be used. The *time* of a pebbling $\mathcal{P} = (\mathbb{P}_0, \ldots, \mathbb{P}_\tau)$ is $\mathsf{time}(\mathcal{P}) = \tau$, and the *space* is $\mathsf{space}(\mathcal{P}) = \max_{0 \le t \le \tau}\{|\mathbb{P}_t|\}$.

We say that a pebbling is *unconditional* if $\mathbb{P}_0 = \emptyset$ and *conditional* otherwise. The *pebbling price* $\mathsf{Peb}_G(\mathbb{P})$ of a pebble configuration $\mathbb{P}$ is the minimum space of any unconditional standard pebbling on G ending in $\mathbb{P}_\tau = \mathbb{P}$, and we define the *reversible pebbling price* $\mathsf{RPeb}_G(\mathbb{P})$ by taking the minimum over all unconditional reversible pebblings reaching $\mathbb{P}$. The pebbling price of a single-sink DAG G with sink z is $\mathsf{Peb}(G) = \mathsf{Peb}_G(\{z\})$, and the reversible pebbling price of G is $\mathsf{RPeb}(G) = \mathsf{RPeb}_G(\{z\})$. We refer to such pebblings as *(complete) pebblings of G* or *pebbling strategies for G*. Again, when G is clear from context we can drop it from the notation, and from now on we will usually abuse notation by omitting the curly brackets around singleton vertex sets.

For technical reasons, we will often be interested in distinguishing particular flavours of reversible pebblings. Suppose that v is a vertex in G and that $\mathcal{P} = (\mathbb{P}_0 = \emptyset, \mathbb{P}_1, \ldots, \mathbb{P}_\tau)$ is a reversible pebbling. We will use the following terminology and notation:

- $\mathcal{P}$ is a *visiting pebbling* of v if $v \in \mathbb{P}_\tau$. The *visiting price* $\mathsf{RPeb}^V(v)$ of v is the minimal space of any such pebbling.
- $\mathcal{P}$ is a *surrounding pebbling* of v if $pred(v) \subseteq \mathbb{P}_\tau$ and the *surrounding price* $\mathsf{RPeb}^S(v)$ is the minimal space of any such pebbling.
- $\mathcal{P}$ is a *persistent pebbling* of v if it is a reversible pebbling of v in the sense defined before, i.e., such that $P_\tau = \{v\}$. We will sometimes refer to $\mathsf{RPeb}(v)$ as the *persistent price* of v to distinguish it from the visiting and surrounding prices.

We also define the visiting price for a single-sink DAG G with sink z as $\mathsf{RPeb}^V(G) = \mathsf{RPeb}^V_G(z)$ and the surrounding price as $\mathsf{RPeb}^S(G) = \mathsf{RPeb}^S_G(z)$.

Note that because of reversibility we could obtain exactly the same visiting space measure by defining a visiting pebbling of v to be a pebbling $\mathcal{P} = (\mathbb{P}_0, \mathbb{P}_1, \ldots, \mathbb{P}_\tau)$ such that $\mathbb{P}_0 = \mathbb{P}_\tau = \emptyset$ and $v \in \bigcup_{0 \le t \le \tau} \mathbb{P}_t$, and let the visiting price be the minimal space of any such pebbling. This is because once we have reached a configuration containing v we can simply run the pebbling backwards (because of reversibility) until we reach the empty configuration again. We can therefore think of a pebbling as *visiting* v if there is a pebble on v at some point but this pebble does not stay on v until the end of the pebbling. In a *persistent* pebbling the pebble remains on v until all other pebbles have been removed. A *surrounding* pebbling, finally, is a pebbling that reaches exactly the point

where a pebble could be placed on v, since all its predecessors are covered by pebbles (i.e., v is "surrounded" by pebbles), but where v is not necessarily pebbled.

It is not hard to see that for a single-sink DAG G we have the inequalities

$$\mathsf{Peb}(G) \leq \mathsf{RPeb}^V(G) \tag{2}$$

and

$$\mathsf{RPeb}^S(G) \leq \mathsf{RPeb}^V(G) \leq \mathsf{RPeb}(G) \ . \tag{3}$$

Perhaps slightly less obviously, we also have the following useful equality.

Proposition 1. *For any vertex v in a DAG G it holds that $\mathsf{RPeb}^S(v) = \mathsf{RPeb}(v) - 1$.*

Proof: To see that $\mathsf{RPeb}(v) \leq \mathsf{RPeb}^S(v) + 1$ consider a surrounding pebbling $\mathcal{P}^S$ of space $\mathsf{RPeb}^S(v)$. Let $\mathcal{P}^*$ be the pebbling which first runs $\mathcal{P}^S$ to surround v, then puts a pebble on v, and finally runs the reverse of $\mathcal{P}^S$ to "unsurround" v (while keeping the pebble on v). Since $\mathcal{P}^*$ is a persistent pebbling of space $\mathsf{RPeb}^S(v)+1$, the inequality follows.

We now prove that $\mathsf{RPeb}^S(v) \leq \mathsf{RPeb}(v) - 1$. Consider a persistent pebbling $\mathcal{P}$ for v of space $\mathsf{RPeb}(v)$. Let t be the last time that a pebble is put on v. Then vertex v is surrounded at time t, and there is a pebble on v since time t. Let $\mathcal{P}_{\geq t}$ be the conditional pebbling obtained from $\mathcal{P}$ after time t, with the modification that vertex v has no pebble throughout $\mathcal{P}_{\geq t}$, and let $\mathcal{P}^R_{\geq t}$ be this pebbling run in reverse. Then $\mathcal{P}^R_{\geq t}$ is a surrounding pebbling in space at most $\mathsf{RPeb}(v) - 1$, and the inequality follows. ∎

D. The Dymond–Tompa and Raz–McKenzie Games

As described above, the *Dymond–Tompa game* on a single-sink DAG G is played in rounds by two players *Pebbler* and *Challenger*. In the first round Pebbler places a pebble on the sink z and Challenger challenges this vertex. In all subsequent rounds, Pebbler places a pebble on an arbitrary empty vertex and Challenger chooses to either challenge this new vertex (which we refer to as *jumping*) or to re-challenge the previously challenged vertex (referred to as *staying*). The game ends when at the end of a round all the (immediate) predecessors of the currently challenged vertex are covered by pebbles.[2] The *Dymond–Tompa price* $\mathsf{DT}(G)$ of G is the maximal number of pebbles r needed for Pebbler to finish the game, or expressed differently the smallest number r such that Pebbler has a strategy to make the game end in at most r rounds regardless of how Challenger plays.

Let us also for completeness describe the *Raz–McKenzie game*, which is also played on a single-sink DAG G by two players *Pebbler* and *Colourer*. In the first round Pebbler places a pebble on the sink z and Colourer colours it red. In all subsequent rounds, Pebbler places a pebble on an arbitrary empty vertex and Colourer then colours this new pebble either red or blue. The game ends when there is a vertex with a red pebble, while all its predecessors in the graph have blue pebbles. The *Raz–McKenzie price* $\mathsf{RM}(G)$ of G is the smallest number r such that Pebbler has a strategy to make the game end in at most r rounds regardless of how Colourer plays.

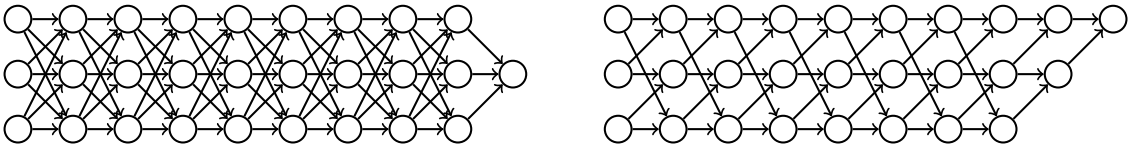
The intuition for this game is that the vertices on the graphs have assigned values true (blue) or false (red), with the condition that each vertex has value equal to the conjunction of the values of its predecessors. Colourer claims that the sink is false, but the above condition vacuously implies that all source vertices must be true. Colourer loses when Pebbler discovers a violation of the condition. Pebbler wants to find the violation as soon as possible, while Colourer wants to fool Pebbler for as long as possible.

In [5] the first author proved that the equalities

$$\mathsf{DT}(G) = \mathsf{RM}(G) = \mathsf{RPeb}(G) \tag{4}$$

hold for any single-sink DAG G, i.e., that the reversible pebbling price, the Dymond–Tompa price and the Raz–McKenzie price all coincide. Thus, any result we prove for one of these games is also guaranteed to hold

[2]We remark that our description follows [5] and thus differs slightly from the original definition in [13], but the two versions are equivalent for all practical purposes.



(a) Path blown up to sequence of $K_{3,3}$-graphs. (b) Road graph of length 9 and width 3.

Figure 1. Modifications of path graphs to amplify difference between reversible and standard pebbling price.

for the other games. The above equalities are very convenient in that they allow us to switch back and forth between the reversible pebble game and the Dymond–Tompa game (or Raz–McKenzie game) when proving upper and lower bounds, depending on which perspective is more suitable at any given time. In particular, when proving lower bounds for reversible pebblings it is often helpful to do so by devising good Challenger strategies in the Dymond–Tompa game. One final technical remark in this context is that in all such strategies that we construct it holds that Challenger will either stay or jump to an ancestor of the currently challenged vertex. Because of this we can assume without loss of generality that Pebbler only pebbles vertices in the subgraph consisting of ancestors of the currently challenged vertex. If Pebbler pebbles some vertex outside of this subgraph Challenger will just stay put on the current vertex, and so Pebbler just wastes a round.

III. Overview of Results and Sketches of Proofs

In this section we give a detailed overview of our results and also sketch some of the main ideas in the proofs. We refer to the upcoming full-length version for all formal proofs and missing technical details.

A. Separation Between Standard and Reversible Pebbling

As mentioned in Section I, the strongest separation hitherto known between standard and reversible pebbling is for the length-ℓ path on vertices $\{v_1, v_2, \ldots, v_{\ell+1}\}$ with edges (v_i, v_{i+1}) for all $i \in [\ell]$, which has a standard pebbling with 2 pebbles whereas reversible pebblings require space $\Theta(\log \ell)$ [3], [22]. We give a simple construction improving this to a multiplicative logarithmic separation.

Theorem 2. *For any function $s(n) = \mathrm{O}\big(n^{1/2-\epsilon}\big)$ for $\epsilon > 0$ constant there are DAGs $\{G_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ with a single sink and fan-in 2 such that $\mathsf{Peb}(G) = \mathrm{O}(s(n))$ and $\mathsf{RPeb}(G) = \Omega(s(n) \log n)$ (where the hidden constant depends linearly on ϵ).*

A first observation is that if we did not have the bounded fan-in restriction, Theorem 2 would be very easy. In such a case we could just take the path of length ℓ, blow up every vertex v_i to s vertices $v_i^1, \ldots, v_i^s$, and add edges $\big(v_i^j, v_{i+1}^{j'}\big)$ for all $j, j' \in [s]$, so that we get a sequence of complete bipartite graphs $K_{s,s}$ glued together as shown in Figure 1(a). It is not hard to show that any reversible pebbling of this DAG would have to do s parallel, synchronized pebblings of the paths $\big\{v_1^j, v_2^j, \ldots, v_{\ell+1}^j\big\}$ for $j \in [s]$, which would require space $\Omega(s \log \ell)$, whereas a standard pebbling would clearly only need space $\mathrm{O}(s)$.

For bounded indegree it is not a priori clear what to do, however, or indeed whether there should even be a multiplicative separation. But it turns out that one can actually simulate a lower bound proof along the same lines as above by considering a layered graph as in Figure 1(b), with s parallel paths of length up to ℓ and with every path having an extra edge fanning out to its "neighbour path" above (or at the bottom for the top row) at each level. We will refer to this construction as a *road graph* of length ℓ and width s (where a path is a maximally narrow road of width 1). It is easy to verify that the standard pebbling price of a road of width $s \geq 2$ is $s + 2$. We claim that the reversible pebbling price is $\Omega\big(s \log(\ell/s)\big)$, from which Theorem 2 follows.

To prove the reversible pebbling lower bound it is convenient to think instead in terms of Challenger strategies in the Dymond–Tompa game. The idea is that Challenger will stay put on the sink until Pebbler has pebbled enough vertices so that there are no pebble-free paths from any source vertex to the sink. Intuitively, the cheapest

way for Pebbler to disconnect the graph is with a straight cut over some layer. When this happens, Challenger looks at the latest pebbled vertex and compares the subgraph between the sources and the cut with the subgraph between the cut and the sink. If more than half of the graph is before the cut, Challenger jumps to the latest pebbled vertex. If not, Challenger stays on the sink. This strategy is then repeated on a graph of at least half the length. Since every cut by Pebbler requires s pebbles, Challenger can survive for roughly $s \log \ell$ rounds (except that the rigorous argument is not quite this simple, and the slightly smaller factor $\log(\ell/s)$ in the formal statement of the theorem is in fact inherent).

B. PSPACE-Completeness of Reversible Pebbling

Moving on to technically more challenging material, let us next discuss our PSPACE-completeness result for reversible pebbling, which we restate here more formally for the record.

Theorem 3. *Given a single-sink DAG G of fan-in 2 and a parameter s, it is* PSPACE*-complete to decide whether G can be reversibly pebbled in space s or not. In more detail, given a QBF $\phi = Q_1 x_1\, Q_2 x_2 \ldots Q_n x_n\, F$, where F is a 3-CNF formula over variables $x_1, \ldots, x_n$, there is a polynomial-time constructible single-sink graph $G(\phi)$ of fan-in 2 and a polynomial-time computable number $\gamma(\phi)$ such that $\mathsf{RPeb}\big(G(\phi)\big) = \gamma(\phi) + [\![\phi \text{ is FALSE}]\!]$.*

At a high level, our proof is similar to that in [16] for standard pebbling: we build gadgets for variables, clauses, and universal and existential quantifiers, and then glue them together in the right way so that pebbling through the gadgets corresponds to verifying satisfying assignments for universally and existentially quantified subformulas of the QBF ϕ. However, the execution of this simple idea is highly nontrivial even in [16], and we run into several additional technical difficulties when we want to do an analogous reduction for reversible pebbling.

For starters, since the difference in pebbling price for graphs $G(\phi)$ obtained from true and false QBFs ϕ is just an additive 1, we need exact control over the pebbling price of all components used in the reduction. For standard pebbling there is no problem here—exact bounds on pebbling price are known for quite a wide selection of graphs—but in the reversible setting this becomes an issue already for almost the simplest possible graph: the complete binary tree of height h. An easy inductive argument shows that the standard pebbling price of such a tree is exactly $h+2$. Since reversible pebblings find paths more challenging than do standard pebblings, one could perhaps expect an extra additive $\log h$ or so in the reversible pebbling bound. However, the asymptotically correct bound turns out to be $h + \Theta(\log^* h)$ as shown in [19], and the upper and lower bounds on the multiplicative constant obtained in that paper are far from tight.

The story is even worse for the workhorse of the construction in [16] (and many other pebbling results), namely *pyramids* of height h, which have i vertices at level i for $i = 1, \ldots, h+1$, and where the jth vertex at level i has incoming edges from the jth and $(j+1)$st vertices at level $i+1$. There is a very neat proof in [10] that the standard pebbling price is again exactly $h+2$, but for reversible pebbling price nothing has been known except that it has to be somewhere between $h+2$ and $h + \mathrm{O}(\log^* h)$ (where the latter bound follows since any strategy for a complete binary tree of height h works for any DAG of height h). As a crucial first step towards establishing Theorem 3, we exactly determine the reversible pebbling price of pyramids (and also binary trees).

Theorem 4. *For Δ denoting a positive integer, let g be the function defined recursively as*

$$g(\Delta) = \begin{cases} 0 & \text{if } \Delta = 1; \\ 2^{g(\Delta-1)+\Delta-2} + g(\Delta-1) & \text{otherwise;} \end{cases}$$

and let the inverse g^{-1} of this function be defined as

$$g^{-1}(h) = \min\{\Delta \mid g(\Delta) \geq h\} \ .$$

Then the persistent pebbling price of a pyramid of height h, as well as of a complete binary tree of height h, is $h + g^{-1}(h)$, where g^{-1} is efficiently computable.

Even though Theorem 4 is an important step, we immediately run into new problems when trying to use it as a building block in our reduction for reversible pebbling. In the standard pebble game a complete pebbling is any pebbling that reaches the sink. For the reversible game there is a subtle distinction in that we can ask whether it is sufficient to just reach the sink or whether the rest of the graph must also be cleared of pebbles. As discussed in Section II, this leads to two different flavours of reversible pebblings, namely *persistent pebblings*, which leave a pebble on the sink with the rest of the graph being empty, and *visiting pebblings*, which just reach the sink (and can then be thought to run in reverse after having visited the sink to clear the whole graph including the sink from pebbles). The pebblings we actually care about are the persistent ones, but we cannot rule out the possibility that subpebblings of gadgets are visiting pebblings. Clearly, the difference in pebbling space is at most 1, but this is exactly the additive 1 of which we cannot afford to lose control! To make things worse, for pyramids it turns out that persistent and visiting pebbling prices actually do differ except in very rare cases.

Because of this, we have to build more involved graph gadgets for which we can guarantee that visiting and persistent prices coincide. These gadgets are constructed in two steps. First, we take a pyramid and append a path of suitable length, depending on the height of the pyramid, to the pyramid sink, resulting in a graph that we call a *teabag*. Second, we take such teabags of smaller and smaller size and stack them on top of one another, which yields a graph that looks a bit like a *Christmas tree*. These Christmas tree graphs are guaranteed to have the same pebbling price regardless of whether a reversible pebbling is visiting or persistent.

With this in hand we are almost ready to follow the approach in the PSPACE-completeness reduction for standard pebbling in [16]. The idea is that we want to build gadgets for the quantifiers in a formula $\phi = \forall x \exists y \cdots Q z\, F$ of specified pebbling price so that the only way to pebble the graph $G(\phi)$ without using too much space is to first pebble the gadget for $\forall x$, then $\exists y$, et cetera, in the correct order until all quantifier gadgets have been pebbled. Once we get to the clause gadgets, we would like that the pebbles in the quantifier gadgets are locked in place encoding a truth value assignment to the variables, and that the only way to pebble through the clause gadgets without exceeding the space budget is if every clause contains at least one literal satisified by this truth value assignment.

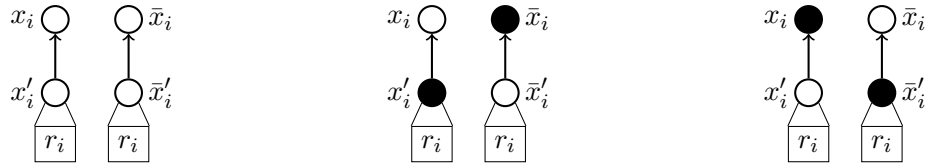
In order to realize this plan, there remains one more significant technical obstacle to overcome, however. To try to explain what the issue is, we need to discuss the PSPACE-completeness reduction in [16] in slightly more detail. The way this reduction imposes an order in which the quantifier gadgets have to be pebbled is that pyramid graphs are included "at the bottom" of the gadgets (i.e., topologically first in order). The source vertices of the quantifier gadgets all appear in such pyramids, and one has to pebble through these pyramids to reach the rest of a gadget (where pebble placements encode variable assignments as mentioned above).

In the first, outermost quantifier gadget the pyramids have large height. In the second gadget the pyramid heights are slightly smaller, et cetera, down to the last, innermost quantifier gadget where the pyramids have smallest height. In this way, the pyramids are used to "lock up" pebbles and force a strict order of pebbling of the gadgets. It can be shown that in order not to exceed the pebbling space budget, any pebbling strategy has to start by pebbling the highest pyramids in the first gadget. If the pebbling starts anywhere else in the graph, this will mean that there are already pebbles elsewhere in the graph when the pebbling strategy reaches the first, highest pyramids in the outermost quantifier, but if so the overall pebbling has to use up too much space to pebble through this pyramid. One can also show that once the pyramids in the outermost quantifier gadget have been pebbled, the pebbling cannot proceed until the next quantifier gadget is pebbled. The pyramids in this gadget have smaller height, but there are also pebbles stuck in place in the outermost gadget, meaning that pyramids must again be pebbled in exactly the right order to stay within the space budget.

These properties can be used to *normalize* pebbling strategies in the standard pebble game. Without loss of generality, one can assume that any strategy that starts pebbling a pyramid in a gadget will complete this local pebbling in one go, leaving a pebble at the sink of the pyramid, and will not place pebbles anywhere else until the pebbling of the pyramid has been completed. Also, once a pyramid in a quantifier gadget has been pebbled in this way, one can prove that it will never be pebbled again since there is now at least one additional pebble at some vertex later in the topological order in the graph, and a repeated pebbling of the pyramid in question

(a) Christmas tree. (b) Turnpike.

Figure 2. Legend for technical gadget building blocks.



(a) Variable gadget. (b) FALSE position. (c) TRUE position.

Figure 3. Gadget for variable x_i and pebble positions corresponding to truth value assignments.

would therefore exceed the space budget. Thus, not only do the pyramids enforce that the gadgets are pebbled in the right order—they also serve as single-entry access points to the gadgets, making sure that each gadget is pebbled exactly once.

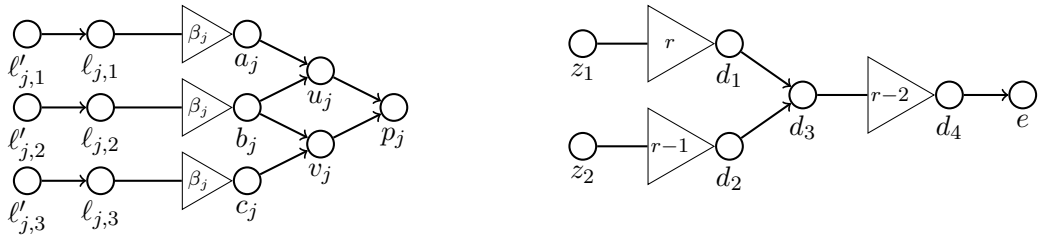
There is no hope of building gadgets with such properties in a reversible pebbling setting. It is simply not true that a reversible pebbling will pebble through a subgraph and then never return. Instead, as already discussed reversible pebblings will proceed in alternating phases of interleaved "forward sweeps" and "reverse sweeps," and subgraphs will be entered also in reverse topological order. Therefore, it is not sufficient to add "space-locking" subgraphs at the source vertices of the gadgets. Rather, we have to insert "single-passage points" inside and in between the gadgets for quantifiers and clauses. We obtain such subgadgets by further tweaking our Christmas tree construction so that it can also connect two vertices in such a way that any pebbling has to "pay a toll" to go through this subgraph. We cannot describe these gadgets, which we call *turnpikes*, in detail here, but mention that the "space-locking" property that they have is that when the entrance vertex is eliminated by having a pebble placed on that vertex, then the cost of pebbling through the rest of the turnpike drops by 1. This is critically used in the subgraph compositions described next.

Assuming the existence of the necessary technical subgraph constructions sketched above, we can now describe the overall structure of our reduction from quantified Boolean formulas to reversible pebbling (where all parameters shown in the figures are fixed appropriately in the formal proofs). In the following figures we denote a Christmas tree of (visiting and persistent) pebbling price r by the symbol in Figure 2(a), where we only display the sink vertex. We denote the turnpike gadget just discussed by the symbol in Figure 2(b). We write r to denote the *toll* parameter of the turnpike, where a turnpike with toll r has persistent price $r + 2$, but only $r + 1$ if we do not count the source a as part of the turnpike.

For every variable x_i we have a *variable gadget* as shown in Figure 3(a), where we think of a truth value assignment ρ as represented by pebbles on vertices $\{\bar{x}_i, x'_i\}$ when $\rho(x_i) = \text{FALSE}$ and on $\{x_i, \bar{x}'_i\}$ when $\rho(x_i) = \text{TRUE}$, as shown in Figures 3(b) and 3(c), respectively.

For every clause C_j we have a *clause gadget* as depicted in Figure 4(a). The vertices labelled $\ell'_{j,k}$ and $\ell_{j,k}$ in Figure 4(a) are identified with the corresponding vertices for the positive or negative literal $\ell_{j,k}$ in the variable gadget in Figure 3(a). If ρ satisfies a literal, then there is a pebble on the entrance vertex of the corresponding turnpike, meaning that we can pebble through the gadget for a clause containing that literal with one less pebble than if ρ does not satisfy the clause.

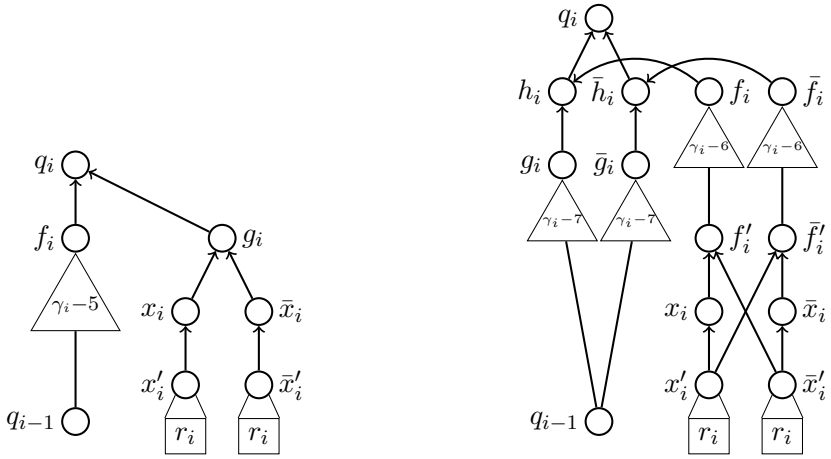
To build the subgraph corresponding to a 3-CNF formula $F = \bigwedge_{j=1}^{m} C_j$ we join clause gadgets sequentially



(a) Clause gadget.

(b) Conjunction gadget.

Figure 4. Gadgets for clauses and CNF formulas.



(a) Existential quantifier gadget.

(b) Universal quantifier gadget.

Figure 5. Quantifier gadgets for variable x_i.

using the *conjunction gadget* in Figure 4(b). For technical reasons we start by joining a dummy graph with the first clause gadget, then we join the result to the second clause gadget, and so on up to the mth clause of F. The resulting graph has the property that if pebbles are placed on the variable gadgets according to an assignment ρ that satisfies F, then the number of additional pebbles needed to pebble the graph is one less than if the assignment is falsifying.

Finally we have one *quantifier gadget* for each variable. To describe this part of the construction, we sort the variables indices in reverse order from the innermost to the outermost quantifier and denote by ϕ_i the subformula with just the i innermost quantifiers, so that $\phi_0 = F = \bigwedge_{j=1}^{m} C_j$, $\phi_i = Q_i x_i \, \phi_{i-1}$ for $Q_i \in \{\forall, \exists\}$, and $\phi = \phi_n$. We construct graphs $G^{(i)} := G(\phi_i)$, starting with $G^{(0)}$ which is just the subgraph corresponding to the CNF formula F. To construct $G^{(i+1)}$ from $G^{(i)}$ we add an existential gadget as in Figure 5(a) if x_i is existentially quantified and a universal gadget as in Figure 5(b) if x_i is universally quantified. An example of the full construction can be found in Figure 6.

Given this construction we argue along the same lines as in in [16], although as mentioned above there are numerous additional technical complications that we cannot elaborate on in this brief overview of the proof. We show that given an assignment ρ_i to $\{x_n, \ldots, x_{i+1}\}$, the number of additional pebbles needed to pebble $G^{(i)}$ differs by 1 depending on whether ϕ_i is true under the assignment ρ_i or not. An existential gadget can be optimally pebbled by setting x_i to any value that satisfies ϕ_{i-1}. To pebble a universal gadget one needs to assign x_i to some value, pebble through the gadget, unset x_i and assign it to the opposite value, and finally pebble through

the gadget again, and both assignments to x_i must yield satisfying assignments to ϕ_{i-1} in order for the pebbling not to go over budget. Proceeding by induction, we establish that the complete graph $G^{(n)}$ can be pebbled within the specified space budget only if $\phi = \phi_n$ is true, which yields Theorem 3.

C. PSPACE-Inapproximability up to Additive Constants

Let us conclude the detailed overview of our contributions by describing what is arguably the strongest result in this paper, namely a strengthening of the PSPACE-completeness of standard pebbling in [16] and of reversible pebbling in Theorem 3 to PSPACE-hardness results for approximating standard and reversible pebbling price to within any additive constant K.

Theorem 5. *For any fixed positive integer K it is* PSPACE*-complete to decide whether a single-sink DAG G with fan-in 2 has (standard or reversible) pebbling price at most s or at least $s + K$.*

We remark that it would of course have been even nicer to prove multiplicative hardness results. We want to stress again, though, that to the best of our knowledge these are the first results ever for hardness of approximation of pebble games in a general setting. The fact that these results hold even for PSPACE could perhaps be taken both as an indication that it should be possible to prove much stronger hardness results for algorithms limited to polynomial time, and as a challenge to do so.

We obtain Theorem 5 by defining and analyzing two graph product constructions, one for standard and one for reversible pebbling, which take two graphs and output product graphs with pebbling price equal to the sum of the pebbling prices of the two input graphs (except for an additive adjustment). These graph products can then be applied iteratively $K - 1$ times to the graphs obtained by the reductions from QBFs. In the next theorem we state the formal properties of these graph products.

Theorem 6. *Given single-sink DAGs G_i of fan-in 2 and size n_i for $i = 1, 2$, there are polynomial-time constructible single-sink DAGs $\mathcal{S}(G_1, G_2)$ and $\mathcal{R}(G_1, G_2)$ of fan-in 2 and size $\mathrm{O}\big((n_1 + n_2)^2\big)$ such that*

- *For standard pebbling price it holds that $\mathsf{Peb}(\mathcal{S}(G_1, G_2)) = \mathsf{Peb}(G_1) + \mathsf{Peb}(G_2) - 1$.*
- *For reversible pebbling price it holds that $\mathsf{RPeb}\big(\mathcal{R}(G_1, G_2)\big) = \mathsf{RPeb}(G_1) + \mathsf{RPeb}(G_2) + 1$.*

In the remainder of this section we try to convey some of the flavour of the arguments used to prove Theorem 6 and to give a sense of some of the technical obstacles that have to be overcome during the analysis. In what follows, we will mostly focus on the reversible pebble game, since it is the technically more challenging and therefore also the more interesting case. We will briefly discuss the product construction for standard pebbling at the very end of the section. We will refer to G_1 as the *outer graph* and G_2 as the *inner graph* in the graph products $\mathcal{R}(G_1, G_2)$ and $\mathcal{S}(G_1, G_2)$.

Intuitively, when taking the graph product of G_1 and G_2 the idea is to replace every vertex v of the outer graph G_1 with a (possibly slightly modified) copy of the inner graph G_2. We will refer to this copy as the v-block in the product graph. The edges inside blocks are specified by the inner graph. For edges $(u, v) \in E(G_1)$ in the outer graph, we will need to connect the sink of the u-block to vertices in the v-block in some way, and this is the crux of the construction.

A first naive approach would be to add an edge from the sink of the u-block to every source vertex of the v-block (as shown in the graph product $\mathcal{N}(G_1, G_2)$ in Figure 7). Sadly, this simple idea fails for both standard and reversible pebbling. It is not hard to find examples showing that the pebbling price of $\mathcal{N}(G_1, G_2)$ is not a function of the pebbling prices of G_1 and G_2.

A slightly more refined idea is to add edges from the sink of the u-block to all vertices in the v-block (as in the graph $\mathcal{T}(G_1, G_2)$ in Figure 7). While we can observe right away that this idea is a non-starter, since it will blow up the fan-in of the product DAG (and with no bounds on fan-in the gap amplification would be trivial), it turns out that the analysis yields interesting insights for the graph product that we will actually use. We will therefore employ this toy construction to showcase some of the ideas and technical challenges that arise in the actual proof of Theorem 6.

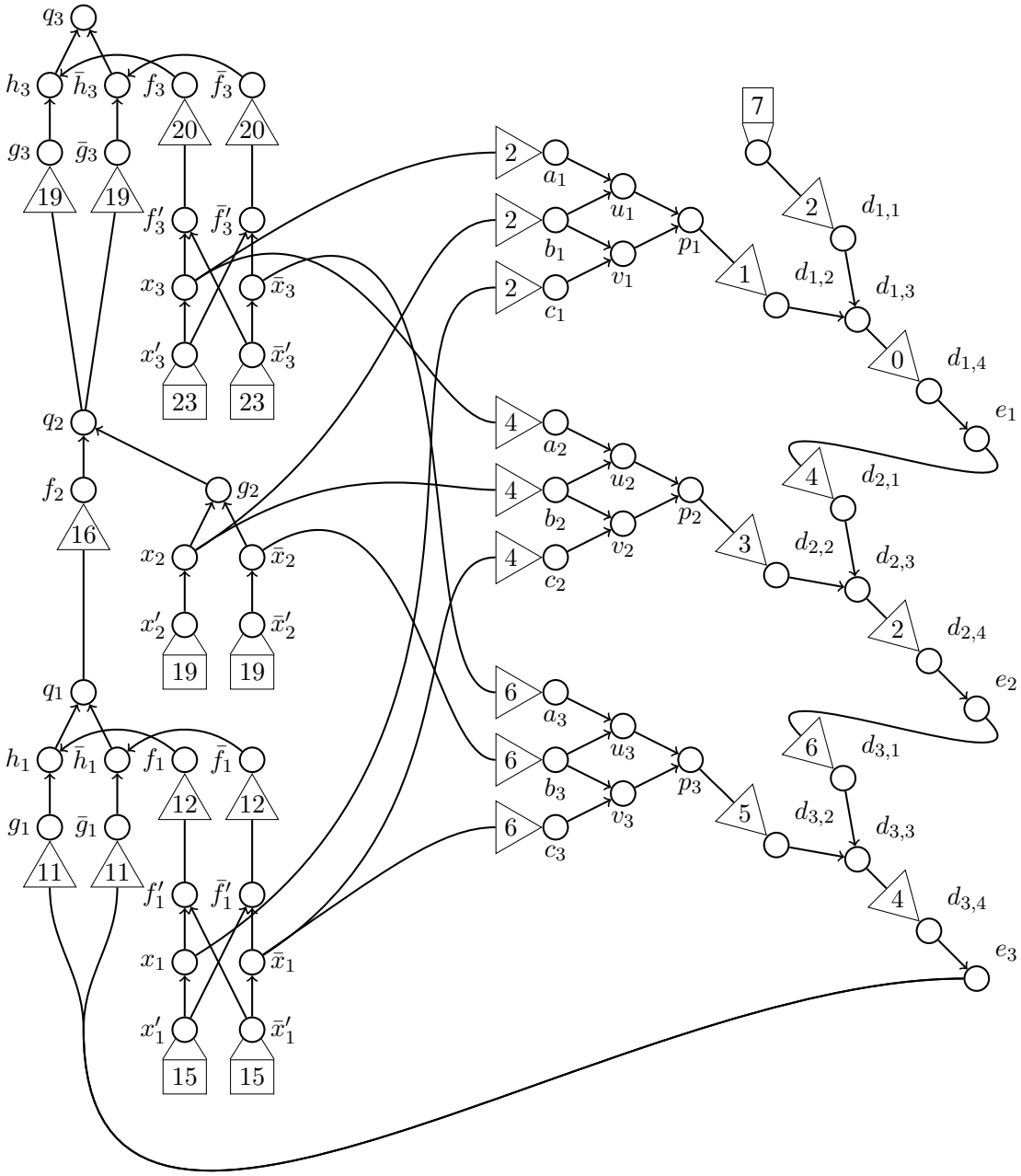


Figure 6. Example of QBF-to-DAG reduction for $\forall x_3 \exists x_2 \forall x_1 (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$.

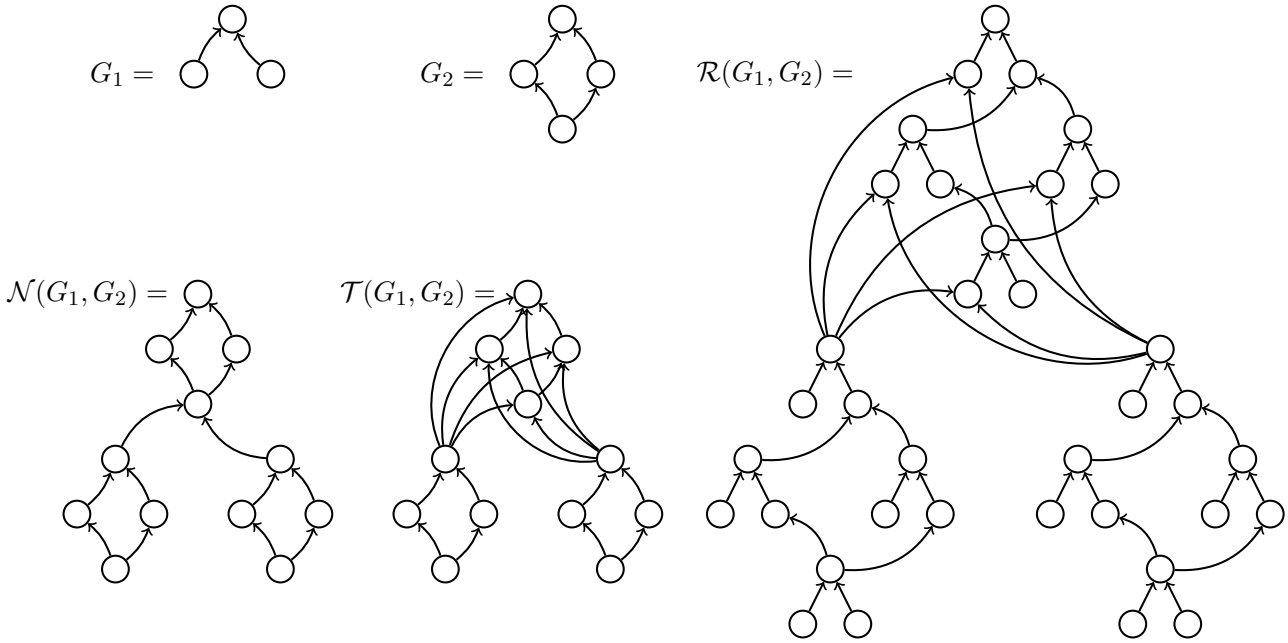


Figure 7. Examples of graph products as applied to a pyramid of height 1 (denoted G_1) and a rhombus (denoted G_2).

Recall that we want to prove that $\mathsf{RPeb}\big(\mathcal{T}(G_1, G_2)\big) = s_1 + s_2 - 1$, where $s_i = \mathsf{RPeb}\big(G_i\big)$ for $i = 1, 2$. To reversibly pebble the product graph $\mathcal{T}(G_1, G_2)$ in at most this amount of space we simulate a minimal space pebbling of G_1, where pebble placement or removal involving a vertex v of G_1 invokes a complete pebbling (or unpebbling) of the copy of G_2 corresponding to the v-block. This simulation uses at most s_2 pebbles in the relevant v-block and at most $s_1 - 1$ pebbles on sinks of other blocks, i.e., no more than $s_1 + s_2 - 1$ pebbles in total.

Proving the lower bound $\mathsf{RPeb}\big(\mathcal{T}(G_1, G_2)\big) \geq s_1 + s_2 - 1$ is the difficult part. Here the approach is to assume that we are given a complete pebbling $\mathcal{P}^{\mathcal{T}}$ of $\mathcal{T}(G_1, G_2)$ and extract from it a pebbling strategy $\mathcal{P}$ for G_1 with the hope that an expensive configuration in $\mathcal{P}$ will also help us to pinpoint an expensive configuration in $\mathcal{P}^{\mathcal{T}}$.

The most straightforward way to obtain a pebbling strategy $\mathcal{P}$ for G_1 from $\mathcal{P}^{\mathcal{T}}$ would be to make a vertex v in G_1 contain a pebble or not depending only on the local pebble configuration of the v-block in $\mathcal{T}(G_1, G_2)$. A natural idea is that v should get a pebble if the v-block has a pebble on its sink and that this pebble should be removed from v when the corresponding block has been emptied of pebbles. If we apply this reduction to a pebbling $\mathcal{P}^{\mathcal{T}}$ of $\mathcal{T}(G_1, G_2)$ we obtain a valid pebbling of G_1. The problem, however, is that $\mathcal{P}^{\mathcal{T}}$ might locally be doing a visiting pebbling (as defined in Section II-C) of the copy of G_2 corresponding to the v-block as a way of moving pebbles on or off other blocks. The consequence of this would be that a configuration of maximal space s_1 in $\mathcal{P}$ may result from a configuration in $\mathcal{P}^{\mathcal{T}}$ that uses space only $s_1 + s_2 - 2$, which is off by one compared to what we need and hence destroys the gap in pebbling price that we are trying to create.

If the visiting price of G_2 is the same as its persistent price, then this problem does not arise, but since this does not hold for graphs in general we need to argue more carefully. It is true that a visiting pebbling of a copy of G_2 might save one pebble as compared to a persistent pebbling, but whenever the sink contains a pebble in a visiting but not persistent pebbling we know that there must also be some other vertex in G_2 that has a pebble (or else the pebbling would be persistent by definition). We need to count such pebbles also in our analysis.

To this end, we make a distinction between blocks that have paid the persistent price and the blocks that have paid the visiting price but not the persistent price. We say that the copy of G_2 corresponding to some v-block

is *visiting-locked*, or just *v-locked* for brevity, at some point in time if the current pebble configuration on its vertices requires reversible pebbling space $s_2 - 1$ to be reached, and that the v-block is *persistent-locked* (or *p-locked* for short) if the configuration has reversible pebbling price s_2.

We can now define a more refined way of projecting $\mathcal{P}^{\mathcal{T}}$-configurations to $\mathcal{P}$-configurations as follows. If a v-block has paid the persistent price, we put a pebble on the corresponding vertex v in G_1. If a block has paid just the visiting price but not the persistent price, then we might still put a pebble on v in G_1, but we only do so if an additional (and slightly delicate) technical condition[3] holds for the pebbling configurations in the blocks corresponding to predecessors of v. This technical condition is designed so that with some additional work[4] we are still able to extract a legal pebbling strategy for G_1 by applying this projection. Furthermore, it will be the case that every pebbling move on a vertex in the outer graph G_1 is the result of the copy of G_2 corresponding to some v-block paying the persistent or visiting price.

The reversible pebbling $\mathcal{P}$ thus extracted will be a persistent pebbling of G_1 by construction, so it must contain a configuration with s_1 pebbles. If this configuration was reached because a block paid the persistent price, then that block contains s_2 pebbles at a time when at least $s_1 - 1$ other blocks have at least 1 pebble each, which is the lower bound that we are after. If the pebble configuration on G_1 in $\mathcal{P}$ was reached because a block paid the visiting price, however, then we are potentially still one pebble short. This is where the additional technical condition mentioned above comes into play. This condition on the predecessor blocks implies that we can find at least one other block that also paid just the visiting price and therefore must contain two pebbles. Summing up, we obtain one block that has at least $s_2 - 1$ pebbles, another block that has at least 2 pebbles, and at least $s_1 - 2$ additional blocks that contain at least 1 pebble each, and so the lower bound holds in this case as well. (Incidentally, this second case is the one where our first, naive, graph product $\mathcal{N}(G_1, G_2)$ fails.)

We already observed, however, that the construction $\mathcal{T}(G_1, G_2)$ does not get us very far because it blows up the indegree of the resulting product graph. Therefore, in the actual proof of Theorem 6 we have to consider a different construction. Briefly, the idea is to start with the graph $\mathcal{T}(G_1, G_2)$ but to bring the indegree down by splitting each vertex w in every block into three vertices $w_{\text{ext}}, w_{\text{int}}, w_{\text{out}}$. All edges to w from other blocks are routed to w_{ext}, all edges from within the block are routed to w_{int}, and finally we add edges from w_{ext} and w_{int} to w_{out}. This is the graph product $\mathcal{R}(G_1, G_2)$ that we use to amplify differences in reversible pebbling price, and that is also illustrated in Figure 7. Now we have to prove that the ideas just outlined work for this new construction where each vertex has been replaced by a small "cloud" of three vertices. The proof of this is much more technically challenging than for the toy case discussed above, and there is no room to go into details here.

At this point we want to switch gears a bit and briefly discuss an application in proof complexity of the PSPACE-hardness result for reversible pebbling. Perhaps the most well-studied proof system for proving the unsatisfiability of, or refuting, CNF formulas is *resolution* (we do not give any formal definition here, referring instead to, for instance, [32] for the necessary details). Every resolution proof can be represented as a DAG, and the *depth* of this proof is the length of a longest path in this DAG. The *resolution depth* of refuting an unsatisfiable CNF formula is the smallest depth of any resolution proof for the formula. It was shown in [5] that computing the reversible pebbling price of a graph of fan-in ℓ reduces to computing the resolution depth of a $(\ell + 1)$-CNF formula, and from this we can obtain the following corollary.

[3]We do not want to get into too detailed a technical argument here, but just for the record pebble configurations on $\mathcal{T}(G_1, G_2)$ can be projected to configurations on G_1 in two stages as follows:

1. Let $\mathbb{P} \subseteq V(G_1)$ consist of all vertices u such that the configuration on the u-block in $\mathcal{T}(G_1, G_2)$ is persistent-locked.
2. Let $\mathbb{P}' \subseteq V(G_1) \setminus \mathbb{P}$ consist of all vertices v such that (a) v is not already surrounded by $\mathbb{P}$, and (b) the configuration on the v-block in $\mathcal{T}(G_1, G_2)$ is visiting-locked.

With this notation, the projected pebble configuration on G_1 is defined to be $\mathbb{P} \cup \mathbb{P}'$.

[4]One added technical complication that we have to take care of here is that when we apply our projection to a pebbling $\mathcal{P}^{\mathcal{T}}$ of $\mathcal{T}(G_1, G_2)$ to obtain a sequence of pebble configurations on G_1, this sequence need not be a valid pebbling of G_1. However, when the projected pebble configuration on G_1 changes after a pebbling move we can insert a legal pebbling sequence between the two projected configurations that passes through all vertices of G_1 corresponding to v-locked blocks, where pebbles are added in topological order and removed in inverse topological order, and this local pebbling does not affect the overall argument.

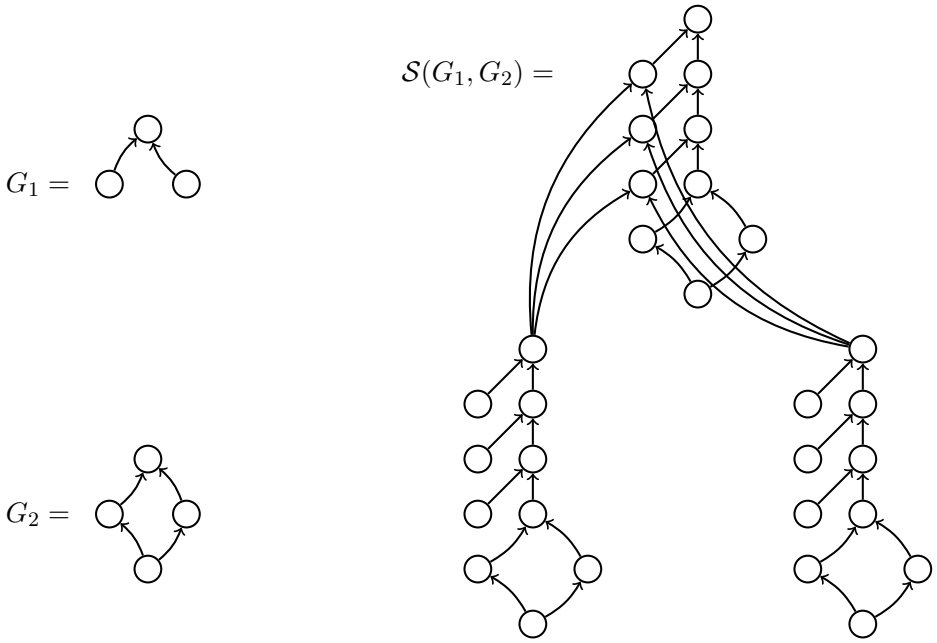


Figure 8. Illustration of standard pebbling graph product $\mathcal{S}(G_1, G_2)$.

Corollary 7. *For any fixed positive integer K, it is* PSPACE*-complete to compute the resolution depth of refuting 3-CNF formulas up to an additive error K.*

Proof: Assuming that we can efficiently compute the resolution depth within an additive error at most K, we show how to efficiently compute the reversible pebbling price of any graph G within an additive error $K+1$, contradicting Theorem 5.

Letting z denote the unique sink of G, we consider a new graph G' which is G augmented with a new successor z' of z (i.e., $G' = (V \cup \{z'\}, E \cup \{(z, z')\})$ in formal notation). The reversible pebbling prices of G and G' differ by at most one. For any graph G, [5] exhibits an efficiently constructible unsatisfiable CNF formula F_G that requires resolution depth equal to the reversible pebbling price of G'. The width of the formula is equal to the fan-in of G plus one, so the result holds for 3-CNFs.

Hence, if we could estimate the resolution depth of refuting F_G, i.e., the reversible pebbling price of G', within error K, this would yield an estimate of the reversible pebbling price of G to within error $K+1$. ■

We wrap up this section by switching back to pebbling and describing the product construction $\mathcal{S}(G_1, G_2)$ used to amplify standard pebbling price. In this construction we also replace every vertex of G_1 with a copy of G_2, but this time we append what we refer to as a *centipede* graph to the sink of every copy. A centipede is a path where each vertex but the source has an extra, unique predecessor. To connect the blocks, for every edge $(u, v) \in E(G_1)$ we add edges from the sink of the u-centipede to every source of the v-centipede. See Figure 8 for an illustration.

Setting $s_i = \mathit{Peb}(G_i)$ for $i = 1, 2$, we can pebble the graph product $\mathcal{S}(G_1, G_2)$ in space $s_1 + s_2 - 1$ by simulating an optimal pebbling of G_1: placing a pebble on a vertex v of G_1 is simulated by optimally pebbling the sink of the corresponding v-block, and removing a pebble is simulated by removing the pebble on the sink.

This pebbling strategy is in fact optimal, and we can show this by projecting any standard pebbling $\mathcal{P}^{\mathcal{S}}$ of $\mathcal{S}(G_1, G_2)$ to a strategy $\mathcal{P}$ for G_1. Each time any block in $\mathcal{S}(G_1, G_2)$ contains s_2 pebbles, we pebble all vertices in G_1 whose predecessors have pebbles and whose corresponding block in $\mathcal{S}$ has a pebble. When a block in $\mathcal{S}(G_1, G_2)$ becomes empty, we remove the pebble from the corresponding vertex in G_1. This projection

has the property that when the sink of a block is pebbled, the corresponding vertex in G_1 is also pebbled. Arguing similarly to in the reversible case, we show that a strategy $\mathcal{P}^{\mathcal{S}}$ for $\mathcal{S}$ using s pebbles yields a strategy for G_1 using $s - s_2 + 1$ pebbles. Therefore, $\mathcal{P}^{\mathcal{S}}$ must use space at least $s_1 + s_2 - 1$, and hence the graph product $\mathcal{S}(G_1, G_2)$ has the property claimed in Theorem 6.

IV. Concluding Remarks

In this paper, we study the pebble game first introduced in [26] as well as the more restricted reversible pebble game in [3], where by [5] the latter game is also equivalent to the Dymond–Tompa game [13] and the Raz–McKenzie game [30].

We establish that it is PSPACE-hard to approximate standard and reversible pebbling price up to any additive constant. To the best of our knowledge, these are the first hardness of approximation results for such pebble games, even for polynomial time. It would be very interesting to show stronger inapproximability results for pebbling price under stronger assumptions. On the one hand, we are only able to show additive hardness, but on the other hand our results hold for arbitrary algorithms using a polynomial amount of memory. It seems reasonable to believe that the problem should become much harder for algorithms restricted to polynomial time, but showing this seems like a challenging task—in some sense, it appears that pebbling might be so hard a problem that it is even hard to prove that it is hard.

Another challenging problem is to prove approximation hardness, or even just PSPACE-completeness, for the *black-white pebble game* [11] modelling nondeterministic computation. This game is a strict generalization of the standard (black) pebble game, and so intuitively it should be at least as hard, but the added option of placing nondeterministic white pebbles anywhere in the graph completely destroys locality and makes the reduction in [16] break down. Hertel and Pitassi [17] showed a PSPACE-completeness result in the nonstandard setting when unbounded (and very large) fan-in is allowed. Essentially, the large fan-in makes it possible to lock down almost all pebbles in one place at a time (namely on the predecessors of a large fan-in vertex to be pebbled) and to completely rule out any use of white pebbles, reducing the whole problem to black pebbling (although this reduction, it should be stressed, is far from trivial). This approach does not work for bounded fan-in graphs, however, which is the standard setting studied in the 1970s and 80s and the setting that could potentially have interesting applications in, for instance, proof complexity.

We also show in this paper that standard black pebbling is asymptotically stronger than reversible pebbling by exhibiting families of DAGs over n vertices which have standard pebblings in space s but for which the reversible pebbling price is $\Omega(s \log n)$. Since any DAG on n vertices with standard pebbling price s can be reversibly pebbled in space $\mathrm{O}(s^2 \log n)$, our separation is at most a linear factor (in $s \leq n$) off from the optimal. It would be interesting to determine how large the separation can be. We do not rule out the possibility that the separation we give might in fact be asymptotically optimal.

Acknowledgements

We are grateful to Anna Gál, Yuval Filmus, Toniann Pitassi, and Robert Robere for stimulating discussions on the topic of pebble games. A special thanks goes to Mladen Mikša, who participated in the initial stages of this work but somehow managed to avoid the pebbling addiction that seized the rest of us…

The first author performed part of this work while at Princeton University. The second, third and fourth authors were funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611. The third author was also supported by Swedish Research Council grants 621-2010-4797 and 621-2012-5645.

References

[1] J. Alwen and V. Serbinenko, "High parallel complexity graphs and memory-hard functions," in *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC '15)*, Jun. 2015, pp. 595–603.

[2] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, Nov. 1973.

[3] ——, "Time/space trade-offs for reversible computation," *SIAM Journal on Computing*, vol. 18, no. 4, pp. 766–776, Aug. 1989.

[4] H. Buhrman, J. Tromp, and P. Vitányi, "Time and space bounds for reversible simulation," *Journal of physics A: Mathematical and general*, vol. 34, pp. 6821–6830, 2001, preliminary version appeared in *ICALP '01*.

[5] S. M. Chan, "Just a pebble game," in *Proceedings of the 28th Annual IEEE Conference on Computational Complexity (CCC '13)*, Jun. 2013, pp. 133–143.

[6] ——, "Pebble games and complexity," Ph.D. dissertation, University of California at Berkeley, 2013.

[7] S. M. Chan and A. Potechin, "Tight bounds for monotone switching networks via Fourier analysis," *Theory of Computing*, vol. 10, pp. 389–419, Oct. 2014, preliminary version appeared in *STOC '12*.

[8] A. K. Chandra, "Efficient compilation of linear recursive programs," in *Proceedings of the 14th Annual Symposium on Switching and Automata Theory (SWAT '73)*, 1973, pp. 16–25.

[9] A. Condon, J. Feigenbaum, C. Lund, and P. W. Shor, "Probabilistically checkable proof systems and nonapproximability of PSPACE-hard functions," *Chicago Journal of Theoretical Computer Science*, vol. 1995, Oct. 1995, preliminary version appeared in *STOC '93*.

[10] S. A. Cook, "An observation on time-storage trade off," *Journal of Computer and System Sciences*, vol. 9, no. 3, pp. 308–316, 1974, preliminary version appeared in *STOC '73*.

[11] S. A. Cook and R. Sethi, "Storage requirements for deterministic polynomial time recognizable languages," *Journal of Computer and System Sciences*, vol. 13, no. 1, pp. 25–37, 1976, preliminary version appeared in *STOC '74*.

[12] C. Dwork, M. Naor, and H. Wee, "Pebbling and proofs of work," in *Proceedings of the 25th Annual International Cryptology Conference (CRYPTO '05)*, ser. Lecture Notes in Computer Science, vol. 3621. Springer, Aug. 2005, pp. 37–54.

[13] P. W. Dymond and M. Tompa, "Speedups of deterministic machines by synchronous parallel machines," *Journal of Computer and System Sciences*, vol. 30, no. 2, pp. 149–161, Apr. 1985.

[14] Y. Filmus, J. Nordström, T. Pitassi, and Y. Wu, Unpublished note, 2010.

[15] Y. Filmus, T. Pitassi, R. Robere, and S. A. Cook, "Average case lower bounds for monotone switching networks," in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS '13)*, Nov. 2013, pp. 598–607.

[16] J. R. Gilbert, T. Lengauer, and R. E. Tarjan, "The pebbling problem is complete in polynomial space," *SIAM Journal on Computing*, vol. 9, no. 3, pp. 513–524, Aug. 1980, preliminary version appeared in *STOC '79*.

[17] P. Hertel and T. Pitassi, "The PSPACE-completeness of black-white pebbling," *SIAM Journal on Computing*, vol. 39, no. 6, pp. 2622–2682, Apr. 2010, preliminary version appeared in *FOCS '07*.

[18] J. Hopcroft, W. Paul, and L. Valiant, "On time versus space," *Journal of the ACM*, vol. 24, no. 2, pp. 332–337, Apr. 1977, preliminary version appeared in *FOCS '75*.

[19] R. Královič, "Time and space complexity of reversible pebbling," *RAIRO – Theoretical Informatics and Applications*, vol. 38, no. 02, pp. 137–161, Apr. 2004.

[20] K.-J. Lange, P. McKenzie, and A. Tapp, "Reversible space equals deterministic space," *Journal of Computer and System Sciences*, vol. 60, no. 2, pp. 354–367, Apr. 2000.

[21] M. Li, J. Tromp, and P. Vitányi, "Reversible simulation of irreversible computation," *Physica D: Nonlinear Phenomena*, vol. 120, pp. 168–176, Sep. 1998.

[22] M. Li and P. Vitányi, "Reversibility and adiabatic computation: Trading time and space for energy," *Proceedings of the Royal Society of London, Series A*, vol. 452, no. 1947, pp. 769–789, Apr. 1996.

[23] A. Lingas, "A PSPACE-complete problem related to a pebble game," in *Proceedings of the 5th Colloquium on Automata, Languages and Programming (ICALP '78)*, 1978, pp. 300–321.

[24] J. Nordström, "Pebble games, proof complexity and time-space trade-offs," *Logical Methods in Computer Science*, vol. 9, pp. 15:1–15:63, Sep. 2013.

[25] ——, "New wine into old wineskins: A survey of some pebbling classics with supplemental results," 2015, manuscript in preparation. Current draft version available at http://www.csc.kth.se/~jakobn/research/.

[26] M. S. Paterson and C. E. Hewitt, "Comparative schematology," in *Record of the Project MAC Conference on Concurrent Systems and Parallel Computation*, 1970, pp. 119–127.

[27] W. J. Paul, R. E. Tarjan, and J. R. Celoni, "Space bounds for a game on graphs," *Mathematical Systems Theory*, vol. 10, pp. 239–251, 1977.

[28] N. Pippenger, "Pebbling," IBM Watson Research Center, Technical Report RC8258, 1980, appeared in Proceedings of the 5th IBM Symposium on Mathematical Foundations of Computer Science, Japan.

[29] A. Potechin, "Bounds on monotone switching networks for directed connectivity," in *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '10)*, Oct. 2010, pp. 553–562.

[30] R. Raz and P. McKenzie, "Separation of the monotone NC hierarchy," *Combinatorica*, vol. 19, no. 3, pp. 403–435, Mar. 1999, preliminary version appeared in *FOCS '97*.

[31] J. E. Savage and S. Swamy, "Space-time tradeoffs for oblivious interger multiplications," in *Proceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP '79)*, 1979, pp. 498–504.

[32] N. Segerlind, "The complexity of propositional proofs," *Bulletin of Symbolic Logic*, vol. 13, no. 4, pp. 417–481, Dec. 2007.

[33] R. Sethi, "Complete register allocation problems," *SIAM Journal on Computing*, vol. 4, no. 3, pp. 226–248, Sep. 1975.

[34] L. J. Stockmeyer and A. R. Meyer, "Word problems requiring exponential time (preliminary report)," in *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC '73)*, 1973, pp. 1–9.

[35] S. Swamy and J. E. Savage, "Space-time trade-offs on the FFT-algorithm," Brown University, Technical Report CS-31, 1977.

[36] ——, "Space-time tradeoffs for linear recursion," *Mathematical Systems Theory*, vol. 16, no. 1, pp. 9–27, 1983.

[37] M. Tompa, "Time-space tradeoffs for computing functions, using connectivity properties of their circuits," in *Proceedings of the 10th annual ACM symposium on Theory of computing (STOC '78)*, 1978, pp. 196–204.

[38] H. Venkateswaran and M. Tompa, "A new pebble game that characterizes parallel complexity classes," *SIAM Journal on Computing*, vol. 18, no. 3, pp. 533–549, Jun. 1989, preliminary version appeared in *FOCS '86*.

[39] R. Williams, "Space-efficient reversible simulations," Cornell University, Tech. Rep., 2000, available at http://web.stanford.edu/~rrwill/spacesim9_22.pdf.

[40] Y. Wu, P. Austrin, T. Pitassi, and D. Liu, "Inapproximability of treewidth and related problems," *Journal of Artificial Intelligence Research*, vol. 49, pp. 569–600, Apr. 2014.