On the Computational Complexity of Minimal Cumulative Cost Graph Pebbling

Jeremiah Blocki[1] and Samson Zhou[2]

[1] Department of Computer Science, Purdue University, West Lafayette, IN.
Email: jblocki@purdue.edu.
[2] Department of Computer Science, Purdue University, West Lafayette, IN.
Email: samsonzhou@gmail.com.

Abstract. We consider the computational complexity of finding a legal black pebbling of a DAG $G = (V, E)$ with minimum cumulative cost. A black pebbling is a sequence $P_0, \ldots, P_t \subseteq V$ of sets of nodes which must satisfy the following properties: $P_0 = \emptyset$ (we start off with no pebbles on G), $\mathsf{sinks}(G) \subseteq \bigcup_{j \leq t} P_j$ (every sink node was pebbled at some point) and $\mathsf{parents}\big(P_{i+1} \backslash P_i\big) \subseteq P_i$ (we can only place a new pebble on a node v if all of v's parents had a pebble during the last round). The cumulative cost of a pebbling $P_0, P_1, \ldots, P_t \subseteq V$ is $\mathsf{cc}(P) = |P_1| + \ldots + |P_t|$. The cumulative pebbling cost is an especially important security metric for data-independent memory hard functions, an important primitive for password hashing. Thus, an efficient (approximation) algorithm would be an invaluable tool for the cryptanalysis of password hash functions as it would provide an automated tool to establish tight bounds on the amortized space-time cost of computing the function. We show that such a tool is unlikely to exist in the most general case. In particular, we prove the following results.

- It is NP-Hard to find a pebbling minimizing cumulative cost.
- The natural linear program relaxation for the problem has integrality gap $\tilde{O}(n)$, where n is the number of nodes in G. We conjecture that the problem is hard to approximate.
- We show that a related problem, find the minimum size subset $S \subseteq V$ such that $\mathsf{depth}(G-S) \leq d$, is also NP-Hard. In fact, under the Unique Games Conjecture there is no $(2-\epsilon)$-approximation algorithm.

1 Introduction

Given a directed acyclic graph (DAG) $G = (V, E)$ the goal of the (parallel) black pebbling game is to start with pebbles on some source nodes of G and ultimately place pebbles on all sink nodes (not necessarily simultaneously). The game is played in rounds and we use $P_i \subseteq V$ to denote the set of currently pebbled nodes on round i. Initially all nodes are unpebbled, $P_0 = \emptyset$, and in each round $i \geq 1$ we may only include $v \in P_i$ if all of v's parents were pebbled in the previous configuration ($\mathsf{parents}(v) \subseteq P_{i-1}$) or if v was already pebbled in the last round ($v \in P_{i-1}$). In the sequential pebbling game we can place at most one

new pebble on the graph in any round (i.e., $|P_i \backslash P_{i-1}| \leq 1$), but in the parallel pebbling game no such restriction applies.

Let $\mathcal{P}_G^{\parallel}$ (resp. $\mathcal{P}_G$) denote the set of all valid parallel (resp. sequential) pebblings of G. We define the cumulative cost (respectively space-time cost) of a pebbling $P = (P_1, \ldots, P_t) \in \mathcal{P}_G^{\parallel}$ to be $\mathsf{cc}(P) = |P_1| + \ldots + |P_t|$ (resp. $\mathsf{st}(P) = t \times \max_{1 \leq i \leq t} |P_i|$), that is, the sum of the number of pebbles on the graph during every round. The *parallel* cumulative pebbling cost of G, denoted $\Pi_{cc}^{\parallel}(G)$ (resp. $\Pi_{st}(G) = \min_{P \in \mathcal{P}_G} \mathsf{st}(G)$), is the cumulative cost of the best legal pebbling of G. Formally,

$$\Pi_{cc}^{\parallel}(G) = \min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{cc}(P) \ , \text{ and} \qquad \Pi_{st}(G) = \min_{P \in \mathcal{P}_G} \mathsf{st}(P) \ .$$

In this paper, we consider the computational complexity of $\Pi_{cc}^{\parallel}(G)$, showing that the value is `NP-Hard` to compute. We also demonstrate that the natural linear programming relaxation for approximating $\Pi_{cc}^{\parallel}(G)$ has a large integrality gap and therefore any approximation algorithm likely requires more powerful techniques.

1.1 Motivation

The pebbling cost of a DAG G is closely related to the cryptanalysis of data-independent memory hard functions (iMHF) [AS15], a particularly useful primitive for password hashing [PHC,BDK16]. In particular, an efficient algorithm for (approximately) computing $\Pi_{cc}^{\parallel}(G)$ would enable us to automate the cryptanalysis of candidate iMHFs. The question is particularly timely as the Internet Research Task Force considers standardizing Argon2i [BDK16], the winner of the password hashing competition [PHC], despite recent attacks [CGBS16,AB16,ABP17] on the construction. Despite recent progress [AB17,ABP17,BZ17] the precise security of Argon2i and alternative constructions is poorly understood.

An iMHF is defined by a DAG G (modeling data-dependencies) on n nodes $V = \{1, \ldots, n\}$ and a compression function H (usually modeled as a random oracle in theoretical analysis)[3] . The label ℓ_1 of the first node in the graph G is simply the hash $H(x)$ of the input x. A vertex $i > 1$ with parents $i_1 < i_2 < \cdots < i_\delta$ has label $\ell_i(x) = H(i, \ell_{i_1}(x), \ldots, \ell_{i_\delta}(x))$. The output value is the label ℓ_n of the last node in G. It is easy to see that any legal pebbling of G corresponds to an algorithm computing the corresponding iMHF. Placing a new pebble on node i corresponds to computing the label ℓ_i and keeping (resp. discarding) a pebble on node i corresponds to storing the label in memory (resp. freeing memory). Alwen and Serbinenko [AS15] proved that in the parallel random oracle model (pROM)

[3] Because the data-dependencies in an iMHF are specified by a static graph, the induced memory access pattern does not depend on the secret input (e.g., password). This makes iMHFs resistant to side-channel attacks. Data-dependent memory hard functions (MHFs) like `scrypt` [Per09] are potentially easier to construct, but they are potentially vulnerable to cache-timing attacks.

of computation, *any* algorithm evaluating such an iMHF could be reduced to a pebbling strategy with (approximately) the same cumulative memory cost.

It should be noted that *any* graph G on n nodes has a sequential pebbling strategy $P \in \mathcal{P}_G$ that finishes in n rounds and has cost $\mathsf{cc}(P) \leq \mathsf{st}(P) \leq n^2$. Ideally, a good iMHF construction provides the guarantee that the amortized cost of computing the iMHF remains high (i.e., $\tilde{\Omega}\left(n^2\right)$) even if the adversary evaluates many instances (e.g.,different password guesses) of the iMHF. Unfortunately, neither large $\Pi_{st}(G)$ nor large $\min_{P \in \mathcal{P}_G^{\parallel}} \mathsf{st}(P)$, are sufficient to guarantee that $\Pi_{cc}^{\parallel}(G)$ is large [AS15]. More recently Alwen and Blocki [AB16] showed that Argon2i [BDK16], the winner of the recently completed password hashing competition [PHC], has much lower than desired amortized space-time complexity. In particular, $\Pi_{cc}^{\parallel}(G) \leq \tilde{O}\left(n^{1.75}\right)$.

In the context of iMHFs, it is important to study $\Pi_{cc}^{\parallel}(G)$, the cumulative pebbling cost of a graph G, in addition to $\Pi_{st}(G)$. Traditionally, pebbling strategies have been analyzed using space-time complexity or simply space complexity. While sequential space-time complexity may be a good model for the cost of computing a single-instance of an iMHF on a standard single-core machine (i.e., the costs incurred by the honest party during password authentication), it does not model the amortized costs of a (parallel) offline adversary who obtains a password hash value and would like to evaluate the hash function on *many* different inputs (e.g., password guesses) to crack the user's password [AS15,AB16]. Unlike $\Pi_{st}(G)$, $\Pi_{cc}^{\parallel}(G)$ models the *amortized* cost of evaluating a data-independent memory hard function on many instances [AS15,AB16].

An efficient algorithm to (approximately) compute $\Pi_{cc}^{\parallel}(G)$ would be an incredible asset when developing and evaluating iMHFs. For example, the Argon2i designers argued that the Alwen-Blocki attack [AB16] was not particularly effective for practical values of n (e.g., $n \leq 2^{20}$) because the constant overhead was too high [BDK16]. However, they could not rule out the possibility that more efficient attacks might exist[4]. As it stands, there is a huge gap between the best known upper/lower bounds on $\Pi_{cc}^{\parallel}(G)$ for Argon2i and for the new DRSample graph [ABH17], since in all practical cases the ratio between the upper bound and the lower bound is at least 10^5. An efficient algorithm to (approximately) compute $\Pi_{cc}^{\parallel}(G)$ would allow us to immediately resolve such debates by automatically generating upper/lower bounds on the cost of computing the iMHF for each running time parameters (n) that one might select in practice. Alwen *et al.* [ABP17] showed how to construct graphs G with $\Pi_{cc}^{\parallel}(G) = \Omega\left(\frac{n^2}{\log n}\right)$. This construction is essentially optimal in theory as results of Alwen and Blocki [AB16] imply that any constant indegree graph has $\Pi_{cc}^{\parallel}(G) = O\left(\frac{n^2 \log\log n}{\log n}\right)$. However, the exact constants one could obtain through

[4] Indeed, Alwen and Blocki [AB17] subsequently introduced heuristics to improve their attack and demonstrated that their attacks were effective even for smaller (practical) values of n by simulating their attack against real Argon2i instances.

a theoretical analysis are most-likely small. A proof that $\Pi^{\parallel}_{cc}(G) \geq \frac{10^{-6} \times n^2}{\log n}$ would be an underwhelming security guarantee in practice, where we may have $n \approx 10^6$. An efficient algorithm to compute $\Pi^{\parallel}_{cc}(G)$ would allow us to immediately determine whether these new constructions provide meaningful security guarantees in practice.

1.2 Results

We provide a number of computational complexity results. Our primary contribution is a proof that the decision problem "is $\Pi^{\parallel}_{cc}(G) \leq k$ for a positive integer $k \leq \frac{n(n+1)}{2}$" is `NP-Complete`.[5] In fact, our result holds even if the DAG G has constant indeg.[6] We also provide evidence that $\Pi^{\parallel}_{cc}(G)$ is hard to approximate. Thus, it is unlikely that it will be possible to automate the cryptanalysis process for iMHF candidates. In particular, we define a natural integer program to compute $\Pi^{\parallel}_{cc}(G)$ and consider its linear programming relaxation. We then show that the integrality gap is at least $\Omega\left(\frac{n}{\log n}\right)$ leading us to conjecture that it is hard to approximate $\Pi^{\parallel}_{cc}(G)$ within constant factors. We also give an example of a DAG G on n nodes with the property that *any* optimal pebbling (minimizing $\Pi^{\parallel}_{cc}$) requires more than n pebbling rounds.

The computational complexity of several graph pebbling problems has been explored previously in various settings [GLT80,HP10]. However, minimizing cumulative cost of a pebbling is a very different objective than minimizing the space-time cost or space. For example, consider a pebbling where the maximum number of pebbles used is significantly greater than the average number of pebbles used. Thus, we need fundamentally new ideas to construct appropriate gadgets for our reduction.[7] We first introduce a natural problem that arises from solving systems of linear equations, which we call `Bounded 2-Linear Covering` (`B2LC`) and show that it is `NP-Complete`. We then show that we can encode a `B2LC` instance as a graph pebbling problem thus proving that the decision version of cummulative graph pebbling is `NP-Hard`.

In Section 5 we also investigate the computational complexity of determining how "depth-reducible" a DAG G is showing that the problem is `NP-Complete` even if G has constant indegree. A DAG G is (e, d)-reducible if there exists a subset $S \subseteq V$ of size $|S| \leq e$ such that $\mathsf{depth}(G - S) < d$. That is, after removing nodes in the set S from G, any remaining directed path has length less than d. If G is not (e, d)-reducible, we say that it is (e, d)-depth robust. It is known that a graph has high cumulative cost (e.g., $\tilde{\Omega}\left(n^2\right)$) if and only if the graph is highly depth robust (e.g., $e, d = \tilde{\Omega}\left(n\right)$) [AB16,ABP17]. Our reduction from

[5] Note that for any G with n nodes we have $\Pi^{\parallel}_{cc}(G) \leq 1 + 2 + \ldots + n = \frac{n(n+1)}{2}$ since we can always pebble G in topological order in n steps if we never remove pebbles.

[6] For practical reasons most iMHF candidates are based on a DAG G with constant indegree.

[7] See additional discussion in Section 3.2.

Vertex Cover preserves approximation hardness.[8] Thus, assuming that $\mathtt{P} \neq \mathtt{NP}$ it is hard to 1.3-approximate e, the minimum size of a set $S \subseteq V$ such that $\mathsf{depth}(G - S) < d$ [DS05]. Under the Unique Games Conjecture [Kho02], it is hard to $(2-\epsilon)$-approximate e for any fixed $\epsilon > 0$ [KR08]. In fact, we show that the linear programming relaxation to the natural integer program to compute e has an integrality gap of $\Omega(n/\log n)$ leading us to conjecture that it is hard to approximate e.

2 Preliminaries

Given a directed acyclic graph (DAG) $G = (V, E)$ and a node $v \in V$ we use $\mathsf{parents}(v) = \{u \ : \ (u, v) \in E\}$ to denote the set of nodes u with directed edges into node v and we use $\mathsf{indeg}(v) = |\mathsf{parents}(v)|$ to denote the number of directed edges into node v. We use $\mathsf{indeg}(G) = \max_{v \in V} \mathsf{indeg}(v)$ to denote the maximum indegree of any node in G. For convenience, we use indeg instead of $\mathsf{indeg}(G)$ when G is clear from context. We say that a node $v \in V$ with $\mathsf{indeg}(v) = 0$ is a source node and a node with no outgoing edges is a sink node. We use $\mathtt{sinks}(G)$ (resp. $\mathtt{sources}(G)$) to denote the set of all sink nodes (resp. source nodes) in G. We will use $n = |V|$ to denote the number of nodes in a graph, and for convenience we will assume that the nodes $V = \{1, 2, 3, \ldots, n\}$ are given in topological order (i.e., $1 \leq j < i \leq n$ implies that $(i, j) \notin E$). We use $\mathsf{depth}(G)$ to denote the length of the longest directed path in G. Given a positive integer $k \geq 1$ we will use $[k] = \{1, 2, \ldots, k\}$ to denote the set of all integers 1 to k (inclusive).

Definition 1. *Given a DAG $G = (V, E)$ on n nodes a legal pebbling of G is a sequence of sets $P = \left(P_0, \ldots, P_t\right)$ such that:*

1. $P_0 = \emptyset$
2. $\forall i > 0$, $v \in P_i \backslash P_{i-1}$ *we have* $\mathsf{parents}(v) \subseteq P_{i-1}$
3. $\forall v \in \mathtt{sinks}(G)$ $\exists 0 < j \leq t$ *such that* $v \in P_j$

The cumulative cost of the pebbling P is $\mathsf{cc}(P) = \sum_{i=1}^{t} |P_i|$, and the space-time cost is $\mathsf{st}(P) = t \times \max_{0 < j \leq t} |P_i|$.

The first condition states that we start with no pebbles on the graph. The second condition states that we can only add a new pebble on node v during round i if we already had pebbles on all of v's parents during round $i-1$. Finally, the last condition states that every sink node must have been pebbled during *some* round.

We use $\mathcal{P}_G^{\parallel}$ to denote the set of all legal pebblings, and we use $\mathcal{P}_G \subset \mathcal{P}_G^{\parallel}$ to denote the set of all sequential pebblings with the additional requirement that $|P_i \backslash P_{i-1}| \leq 1$ (i.e., we place at most one new pebble on the graph during ever

[8] Note that when $d = 0$ testing whether a graph G is (e, d) reducible is *equivalent* to asking whether G has a vertex cover of size e. Our reduction establishes hardness for $d \gg 1$.

round i). We use $\Pi^{\parallel}_{cc}(G) = \min_{P \in \mathcal{P}^{\parallel}_G} \mathsf{cc}(P)$ to denote the cumulative cost of the best legal pebbling.

Definition 2. *We say that a directed acyclic graph (DAG) $G = (V, E)$ is (e, d)-depth robust if $\forall S \subseteq V$ of size $|S| \leq e$ we have $\mathsf{depth}(G - S) \geq d$. If G contains a set $S \subseteq V$ of size $|S| \leq e$ such that $\mathsf{depth}(G - S) \leq d$ then we say that G is (e, d)-reducible.*

Decision Problems

The decision problem `minCC` is defined as follows:
Input: a DAG G on n nodes and an integer $k < n(n+1)/2$. [9]
Output: *Yes*, if $\Pi^{\parallel}_{cc}(G) \leq k$; otherwise *No*.

Given a constant $\delta \geq 1$ we use $\texttt{minCC}_\delta$ to denote the above decision problem with the additional constraint that $\mathsf{indeg}(G) \leq \delta$. It is clear that $\texttt{minCC} \in \texttt{NP}$ and $\texttt{minCC}_\delta \in \texttt{NP}$ since it is easy to verify that a candidate pebbling P is legal and that $\mathsf{cc}(P) \leq k$. One of our primary results is to show that the decision problems `minCC` and $\texttt{minCC}_2$ are `NP-Complete`. In fact, these results hold even if we require that the DAG G has a single sink node.

The decision problem $\texttt{REDUCIBLE}_d$ is defined as follows:
Input: a DAG G on n nodes and positive integers $e, d \leq n$.
Output: *Yes*, if G is (e, d)-reducible; otherwise *No*.

We show that the decision problem $\texttt{REDUCIBLE}_d$ is `NP-Complete` for *all* $d > 0$ by a reduction from Cubic Vertex Cover, defined below. Note that when $d = 0$ $\texttt{REDUCIBLE}_d$ *is* Vertex Cover. We use $\texttt{REDUCIBLE}_{d,\delta}$ to denote the decision problem with the additional constraint that $\mathsf{indeg}(G) \leq \delta$.

The decision problem `VC` (resp. `CubicVC`) is defined as follows:
Input: a graph G on n vertices (`CubicVC`: each with degree 3) and a positive integer $k \leq \frac{n}{2}$.
Output: *Yes*, if G has a vertex cover of size at most k; otherwise *No*.

To show that `minCC` is `NP-Complete` we introduce a new decision problem `B2LC`. We will show that the decision problem `B2LC` is `NP-Complete` and we will give a reduction from `B2LC` to `minCC`.

The decision problem Bounded 2-Linear Covering (`B2LC`) is defined as follows:
Input: an integer n, k positive integers $0 \leq c_1, \ldots, c_k$, an integer $m \leq k$ and k equations of the form $x_{\alpha_i} + c_i = x_{\beta_i}$, where $\alpha_i, \beta_i \in [n]$ and $i \in [k]$. We require that $\sum_{i=1}^{k} c_i \leq p(n)$ for some fixed polynomial n.
Output: *Yes*, if we can find mn integers $x_{y,z} \geq 0$ (for each $1 \leq y \leq m$ and $1 \leq z \leq n$) such that for each $i \in [k]$ there exists $1 \leq y \leq m$ such that $x_{y,\alpha_i} + c_i = x_{y,\beta_i}$ (that is the assignment $x_1, \ldots, x_n = x_{y,1}, \ldots, x_{y,n}$ satisfies the i^{th} equation); otherwise *No*.

[9] See footnote 5.

3 Related Work

The sequential black pebbling game was introduced by Hewitt and Paterson [HP70], and by Cook [Coo73]. It has been particularly useful in exploring space/time trade-offs for various problems like matrix multiplication [Tom78], fast fourier transformations [SS78,Tom78], integer multiplication [SS79b] and many others [Cha73,SS79a]. In cryptography it has been used to construct/analyze proofs of space [DFKP15,RD16], proofs of work [DNW05,MMV13] and memory-bound functions [DGN03] (functions that incur many expensive cache-misses [ABW03]). More recently, the black pebbling game has been used to analyze memory hard functions e.g., [AS15,AB16,ABP17,AT17].

3.1 Password Hashing and Memory Hard Functions

Users often select low-entropy passwords which are vulnerable to offline attacks if an adversary obtains the cryptographic hash of the user's password. Thus, it is desirable for a password hashing algorithm to involve a function $f(.)$ which is moderately expensive to compute. The goal is to ensure that, even if an adversary obtains the value $(username, f(pwd, salt), salt)$ (where $salt$ is some randomly chosen value), it is prohibitively expensive to evaluate $f(., salt)$ for millions (billions) of different password guesses. PBKDF2 (Password Based Key Derivation Function 2) [Kal00] is a popular moderately hard function which iterates the underlying cryptographic hash function many times (e.g., 2^{10}). Unfortunately, PBKDF2 is insufficient to protect against an adversary who can build customized hardware to evaluate the underlying hash function. The cost computing a hash function H like SHA256 or MD5 on an Application Specific Integrated Circuit (ASIC) is dramatically smaller than the cost of computing H on traditional hardware [NBF+15].

[ABW03], observing that cache-misses are more egalitarian than computation, proposed the use of "memory-bound" functions for password hashing — a function which maximizes the number of expensive cache-misses. Percival [Per09] observed that memory costs tend to be stable across different architectures and proposed the use of memory-hard functions (MHFs) for password hashing. Presently, there seems to be a consensus that memory hard functions are the 'right tool' for constructing moderately expensive functions. Indeed, all entrants in the password hashing competition claimed some form of memory hardness [PHC]. As the name suggests, the cost of computing a memory hard function is primarily memory related (storing/retrieving data values). Thus, the cost of computing the function cannot be significantly reduced by constructing an ASIC. Percival [Per09] introduced a candidate memory hard function called `scrypt`, but `scrypt` is potentially vulnerable to side-channel attacks as its computation yields a memory access pattern that is data-dependent (i.e., depends on the secret input/password). Due to the recently completed password hashing competition [PHC] we have many candidate data-independent memory

hard functions such as Catena [FLW13] and the winning contestant Argon2i-A [BDK15].[10]

iMHFs and Graph Pebbling All known candidate iMHFs can be described using a DAG G and a hash function H. Graph pebbling is a particularly useful as a tool to analyze the security of an iMHF [AS15,CGBS16,FLW13]. A pebbling of G naturally corresponds to an algorithm to compute the iMHF. Alwen and Serbinenko [AS15] showed that in the pROM model of computation, *any* algorithm to compute the iMHF corresponds to a pebbling of G.

Measuring Pebbling Costs In the past, MHF analysis has focused on space-time complexity [Per09,FLW13,BCS16]. For example, the designers of Catena [FLW13] showed that their DAG G had high sequential space-time pebbling cost $\Pi_{st}(G)$ and Boneh *et al.* [BCS16] showed that Argon2i-A and their own iMHF candidate iBH ("balloon hash") have (sequential) space-time cost $\Omega(n^2)$. Alwen and Serbinenko [AS15] observed that these guarantees are insufficient for two reasons: (1) the adversary may be parallel, and (2) the adversary might amortize his costs over multiple iMHF instances (e.g., multiple password guesses). Indeed, there are now multiple known attacks on Catena [BK15,AS15,AB16]. Alwen and Blocki [AB16,AB17] gave attacks on Argon2i-A, Argon2i-B, iBH, and Catena with lower than desired amortized space-time cost — $\Pi^{\|}_{cc}(G) \leq O(n^{1.8})$ for Argon2i-B, $\Pi^{\|}_{cc}(G) \leq \tilde{O}(n^{1.75})$ for Argon2i-A and iBH and $\Pi^{\|}_{cc}(G) \leq O(n^{5/3})$ for Catena. This motivates the need to study cumulative cost $\Pi^{\|}_{cc}$ instead of space-time cost since amortized space-time complexity approaches $\Pi^{\|}_{cc}$ as the number of iMHF instances being computed increases.

Alwen *et al.* [ABP17] recently constructed a constant indegree graph G with $\Pi^{\|}_{cc}(G) = \Omega(\frac{n^2}{\log n})$. From a theoretical standpoint, this is essentially optimal as any constant indeg DAG has $\Pi^{\|}_{cc} = O(\frac{n^2 \log\log n}{\log n})$ [AB16], but from a practical standpoint the critically important constants terms in the lower bound are not well understood.

Ren and Devedas [RD17] recently proposed an alternative metric MHFs called bandwidth hardness. The key distinction between bandwidth hardness and cumulative pebbling cost is that bandwidth hardness attempts to approximate *energy costs*, while cumulative pebbling cost attempts to approximate amortized capital costs (i.e., the cost of all of the DRAM chips divided by the number of MHF instances that can be computed before the DRAM chip fails). In this paper we focus on the cumulative pebbling cost metric as we expect amortized capital costs to dominate for sufficiently large n. In particular, bandwidth costs scale linearly with the running time n (at best), while cumulative pebbling costs can scale quadratically with n.

[10] The specification of Argon2i has changed several times. We use Argon2i-A to refer to the version of Argon2i from the password hashing competition, and we use Argon2i-B to refer to the version that is currently being considered for standardization by the Cryptography Form Research Group (CFRG) of the IRTF[BDKJ16].

3.2 Computational Complexity of Pebbling

The computational complexity of various graph pebbling has been explored previously in different settings [GLT80,HP10]. Gilbert *et al.* [GLT80] focused on space-complexity of the black-pebbling game. Here, the goal is to find a pebbling which minimizes the total number of pebbles on the graph at any point in time (intuitively this corresponds to minimizing the maximum space required during computation of the associated function). Gilbert *et al.* [GLT80] showed that this problem is PSPACE complete by reducing from the truly quantified boolean formula (TQBF) problem.

The optimal (space-minimizing) pebbling of the graphs from the reduction of Gilbert *et al.* [GLT80] often require exponential time. By contrast, observe that $\texttt{minCC} \in NP$ because any DAG G with n nodes this algorithm has a pebbling P with $\texttt{cc}(P) \leq \texttt{st}(P) \leq n^2$. Thus, if we are minimizing `cc` or `st` cost, the optimal pebbling of G will trivially never require more than n^2 steps. Thus, we need different tools to analyze the computational complexity of the problem of finding a pebbling with low cumulative cost.

In Appendix D, we show that the optimal pebbling from [GLT80] does take polynomial time if the TQBF formula only uses existential quantifiers (i.e., if we reduce from 3SAT). Thus, the reduction of Gilbert *et al.* [GLT80] can also be extended to show that it is `NP-Complete` to check whether a DAG G admits a pebbling P with $\texttt{st}(P) \leq k$ for some parameter k. The reduction, which simply appends a long chain to the original graph, exploits the fact that if we increase space-usage even temporarily we will dramatically increase `st` cost. However, this reduction does not extend to cumulative cost because the penalty for temporarily placing large number of pebbles can be quite small as we do not keep these pebbles on the graph for a long time.

4 NP-Hardness of `minCC`

In this section we prove that `minCC` is `NP-Complete` by reduction from `B2LC`. Showing that $\texttt{minCC} \in \texttt{NP}$ is straightforward so we will focus on proving that the decision problem is `NP-Hard`. We first provide some intuition about the reduction.

Recall that a `B2LC` instance consists of n variables $x_1, \ldots, x_n$, and k equations of the form $x_{\alpha_i} + c_i = x_{\beta_i}$, where $\alpha_i, \beta_i \in [n]$, $i \in [k]$, and each $c_i \leq p(n)$ is a positive integer bounded by some polynomial in n. The goal is to determine whether there exist m different variable assignments such that each equation is satisfied by *at least one* of the m assignments. Formally, the goal is to decide if there exists a set of $m < k$ variable assignments: $x_{y,z} \geq 0$ for each $1 \leq y \leq m$ and $1 \leq z \leq n$ so that for each $i \in [k]$ there exists $y \in [m]$ such that $x_{y,\alpha_i} + c_i = x_{y,\beta_i}$ — that is the i^{th} equation $x_{\alpha_i} + c_i = x_{\beta_i}$ is satisfied by the y^{th} variable assignment $x_{y,1}, \ldots, x_{y,n}$. For example, if $k = 2$ and the equations are $x_1 + 1 = x_2$ and $x_2 + 2 = x_3$, then $m = 1$ suffices to satisfy all the equations. On the other hand, if $x_1 + 1 = x_2$ and $x_1 + 2 = x_2$, then we require $m \geq 2$ since the equations are no longer independent. Observe that for $m = 1$, `B2LC` seeks a single satisfying

assignment, whereas for $m \geq k$, each equation can be satisfied by a separate assignment of the variables (specifically, the i^{th} assignment is all zeroes except $x_{\beta_i} = c_i$).

Suppose we are given an instance of B2LC. We shall construct a minCC instance $G_{\texttt{B2LC}}$ in such a way that the optimal pebbling of $G_{\texttt{B2LC}}$ has "low" cost if the instance of B2LC is satisfiable and otherwise, has "high" cost. The graph B2LC will be constructed from three different types of gadgets: τ gadgets $C_i^1, \ldots, C_i^\tau$ for each variable x_i, a gadget E_i for each equation and a "m-assignments" gadget M_i for each variable x_i. Here τ is a parameter we shall set to create a gap between the pebbling costs of satisfiable and unsatisfiable instances of B2LC. Each gadget is described in more detail below.

Variable Gadgets Our first gadget is a chain of length $c = \sum c_i$ so that each node is connected to the previous node, and can only be pebbled if there exists a pebble on the previous node in the previous step, such as in Figure 1. For each variable x_i in our B2LC instance we will add τ copies of our chain gadget $C_i^1, \ldots, C_i^\tau$. Formally, for each $j \in [\tau]$ the chain gadget C_i^j consists of c vertices $v_i^{j,1}, \ldots, v_i^{j,c}$ with directed edges $\left(v_i^{j,z}, v_i^{j,z+1}\right)$ for each $z < c$. We will later add a gadget to ensure that we must walk a pebble down each of these chains m different times and that in any optimal pebbling $P \in \mathcal{P}^{\parallel}_{G_{\texttt{B2LC}}}$ (with $\texttt{cc}(P) = \Pi^{\parallel}_{cc}(G_{\texttt{B2LC}})$) the walks on each chain gadget $C_i^1, \ldots, C_i^\tau$ are synchronized e.g., for each pebbling round y and for each $z \leq c$ we have $v_i^{j,z} \in P_y \leftrightarrow \{v_i^{1,z}, \ldots v_i^{\tau,z}\} \subseteq P_y$. Intuitively, each *time* at which we begin walking a pebble down these chains will correspond to an assignment of the B2LC variable x_i. Hence, it suffices to have $c = \sum c_i$ nodes in the chain.

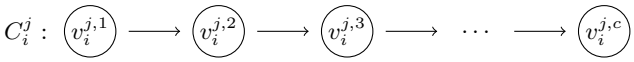

Fig. 1. Example variable gadget C_i^j of length $c = \sum c_i$. $G_{\texttt{B2LC}}$ replicates this gadget τ times: $C_i^1, \ldots, C_i^\tau$. Each of the τ copies behaves the same.

Equation Gadget For the i^{th} equation $x_{\alpha_i} + c_i = x_{\beta_i}$, the gadget E_i is a chain of length $c - c_i$. For each $j \in [\tau]$ we connect the equation gadget E_i to each of the variable gadgets $C_{\alpha_i}^j$ and $C_{\beta_i}^j$ as follows: the a^{th} node e_j^a in chain E_j has incoming edges from vertices $v_{\alpha_i}^{l,a}$ and $v_{\beta_i}^{l,a+c_i}$ for all $1 \leq l \leq \tau$, as demonstrated in Figure 2. To pebble the equation gadget, the corresponding variable gadgets must be pebbled synchronously, distance c_i apart.

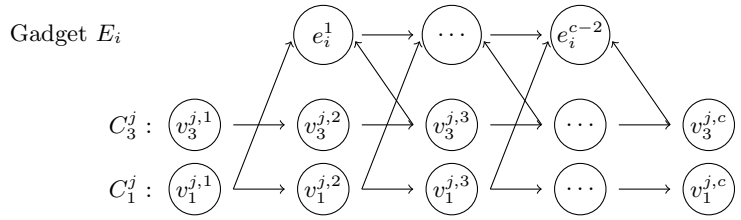

Fig. 2. The gadget E_i for equation $x_3+2 = x_1$. The example shows how E_i is connected to the variable gadgets C_1^j and C_3^j for each $j \in [\tau]$.

Intuitively, if the equation $x_\alpha + c_i = x_\beta$ is satisfied by the j^{th} assignment, then on the j^{th} time we walk pebbles down the chain x_α and x_β, the pebbles on each chain will be synchronized (i.e., when we have a pebble on $v_\alpha^{l,a}$, the a^{th} link in the chain representing x_α we will have a pebble on the node $v_\beta^{l,a+c_i}$ during the same round) so that we can pebble all of the nodes in the equation gadget, such as in Figure 3. On the other hand, if the pebbles on each chain are not synchronized appropriately, we cannot pebble the equation gadget. Finally, we create a single sink node linked from each of the k equation chains, which can only be pebbled if all equation nodes are pebbled.

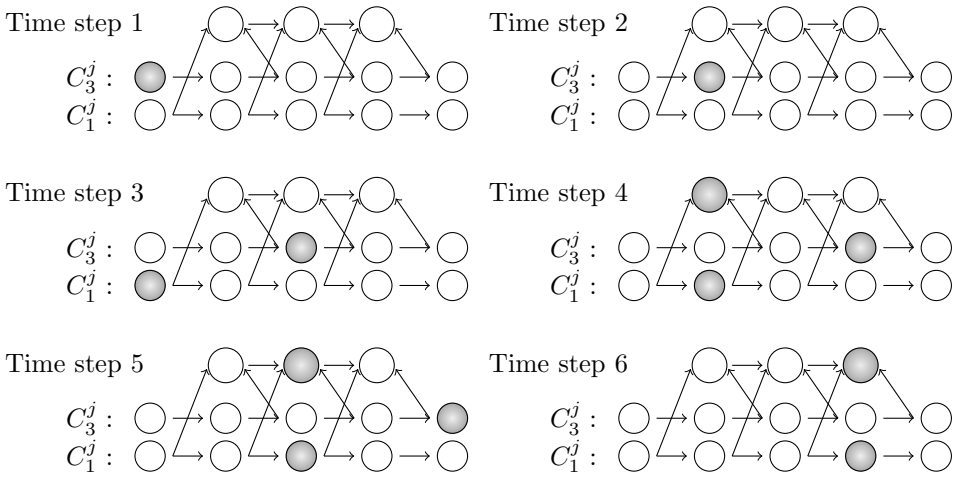

Fig. 3. A pebbling of the equation gadget $x_3 + 2 = x_1$ (at the top) using the satisfying assignment $x_3 = 1$ and $x_1 = 3$.

We will use another gadget, the *assignment gadget*, to ensure that in any legal pebbling, we need to "walk" a pebble down each chain C_i^j on m different times. Each node $v_i^{j,z}$ of a variable gadget in a satisfiable `B2LC` instance has a pebble on

it during exactly m rounds. On the other hand, the assignment gadget ensures that for any unsatisfiable B2LC instance, there exists some $i \leq n$ and $z \leq c$ such that each of the nodes $v_i^{1,z}, \ldots, v_i^{\tau,z}$ are pebbled during at least $m+1$ rounds.

We will tune the parameter τ to ensure that any such pebbling is more expensive, formalized in Claim 8 in Appendix A.

m assignments gadget Our final gadget is a path of length cm so that each node is connected to the previous node. We create a path gadget M_i of length cm for each variable x_i and connect M_i to each the variable gadgets $C_i^1, \ldots C_i^\tau$ as follows: for every node z_i^{p+qc} in the path with position $p+qc > 1$, where $1 \leq p \leq c$ and $0 \leq q < m-1$, we add an edge to z_i^{p+qc} from each of the nodes $v_i^{j,p}$, $1 \leq j \leq \tau$ (that is, the p^{th} node in all τ chains $C_i^1, \ldots C_i^\tau$ representing the variable x_i). We connect the final node in each of the n paths to the final sink node v_{sink} in our graph $G_{\texttt{B2LC}}$.

Intuitively, to pebble v_{sink} we must walk a pebble down each of the gadgets M_i which in turn requires us to walk a pebble along each chain C_i^j, $1 \leq j \leq \tau$, at least m times. For example, see Figure 4.

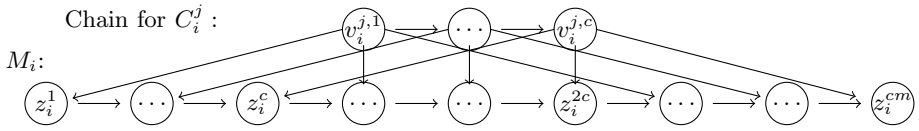

Fig. 4. The gadget M_i for variable x_i is a path of length cm. The example shows how M_i is connected to C_i^j for each $j \in [\tau]$. The example shows $m = 3$ passes and $c = 3$.

Figure 5 shows an example of a reduction in its entirety when $\tau = 1$.

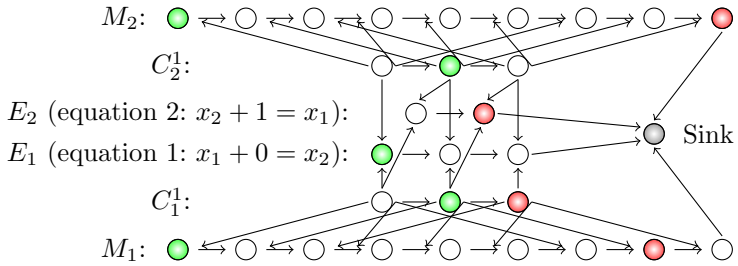

Fig. 5. An example of a complete reduction $G_{\texttt{B2LC}}$, again $m = 3$ and $c = 3$. The green nodes represent the pebbled vertices at time step 2 while the red nodes represent the pebbled vertices at time step 10.

Lemma 1. *If the* B2LC *instance has a valid solution, then* $\Pi^{\parallel}_{cc}(G_{\texttt{B2LC}}) \leq \tau cmn + 2cmn + 2ckm + 1$.

Lemma 2. *If the* B2LC *instance does not have a valid solution, then* $\Pi^{\parallel}_{cc}(G_{\texttt{B2LC}}) \geq \tau cmn + \tau$.

We outline the key intuition behind Lemma 1 and Lemma 2 and refer to the appendix for the formal proofs. Intuitively, any solution to B2LC corresponds to m walks across the τn chains C_i^j, $1 \leq i \leq n$, $1 \leq j \leq \tau$ of length c. If the B2LC instance is satisfiable then we can synchronize each of these walks so that we can pebble every equation chain E_j and path M_j along the way. Thus, the total cost is τcmn plus the cost to pebble the k equation chains E_j ($\leq 2ckm$), the cost to pebble the n paths M_j ($\leq 2cmn$) plus the cost to pebble the sink node 1.

We then prove a structural property about the optimal pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\parallel}_{G_{\texttt{B2LC}}}$. In particular, Claim 8 from the appendix states that if $P = (P_0, \ldots, P_t)$ is optimal (i.e., $\texttt{cc}(P) = \Pi^{\parallel}_{cc}(G_{\texttt{B2LC}})$) then during each pebbling round $y \leq t$ the pebbles on each of the chains $C_i^1, \ldots, C_i^\tau$ are synchronized. Formally, for every $y \leq t$, $i \leq n$ and $z \leq c$ we either have (1) $\left\{v_i^{1,z}, \ldots, v_i^{\tau,z}\right\} \subseteq P_y$, or (2) $\left\{v_i^{1,z}, \ldots, v_i^{\tau,z}\right\} \bigcap P_y = \emptyset$ — otherwise we could reduce our pebbling cost by discarding these unnecessary pebbles.

If the B2LC instance is not satisfied then we must adopt a "cheating" pebbling strategy P, which does not correspond to a B2LC solution. We say that P is a "cheating" pebbling if some node $v_i^{j,z} \in C_i^j$ is pebbled during at least $m+1$ rounds. We can use Claim 8 to show that the cost of *any* "cheating" pebbling is at least $\texttt{cc}(P) \geq \tau\,(mc+1)$. In particular, P must incur cost at least $\tau mc(n-1)$ to walk a pebble down each of the chains $C_{i'}^j$ with $i' \neq i$ and $1 \leq j \leq \tau$. By Claim 8, any cheating pebbling P incurs costs at least $\tau(mc+1)$ on each of the chains $C_i^1, \ldots, C_i^\tau$. Thus, the cumulative cost is at least $\tau cmn + \tau$.

Theorem 1. minCC *is* NP-Complete.

Proof. It suffices to show that there is a polynomial time reduction from B2LC to minCC since B2LC is NP-Complete (see Theorem 3). Given an instance $\mathcal{P}$ of B2LC, we create the corresponding graph G as described above. This is clearly achieved in polynomial time. By Lemma 1, if $\mathcal{P}$ has a valid solution, then $\Pi^{\parallel}_{cc}(G) \leq \tau cmn + 2cmn + 2ckm + 1$. On the other hand, by Lemma 2, if $\mathcal{P}$ does not have a valid solution, then $\Pi^{\parallel}_{cc}(G) \geq \tau cmn + \tau$. Therefore, setting $\tau > 2cmn + 2ckm + 1$ (such as $\tau = 2cmn + 2ckm + 2$) allows one to solve B2LC given an algorithm which outputs $\Pi^{\parallel}_{cc}(G)$.

Theorem 2. $\texttt{minCC}_\delta$ *is* NP-Complete *for each* $\delta \geq 2$.

Note that the only possible nodes in $G_{\texttt{B2LC}}$ with indegree greater than two are the nodes in the equation gadgets $E_1, \ldots, E_m$, and the final sink node. The equation gadgets can have indegree up to $2\tau + 1$, while the final sink node has indegree

$n + m$. To show that $\texttt{minCC}_\delta$ is NP-Complete when $\delta = 2$ we can replace the incoming edges to each of these nodes with a binary tree, so that all vertices have indegree at most two. By changing τ appropriately, we can still distinguish between instances of B2LC using the output of $\texttt{minCC}_\delta$. We refer to the appendix for a sketch of the proof of Theorem 2.

Theorem 3. B2LC *is* NP-Complete.

To show that B2LC is NP-Complete we will reduce from the problem 3-PARTITION, which is known to be NP-Complete. The decision problem 3-PARTITION is defined as follows:
Input: A multi-set S of $m = 3n$ positive integers $x_1, \ldots, x_m \geq 1$ such that (1) we have $\frac{T}{4n} < x_i < \frac{T}{2n}$ for each $1 \leq i \leq m$, where $T = x_1 + \ldots + x_m$, and (2) we require that $T \leq p(n)$ for a fixed polynomial p.[11]
Output: *Yes,* if there is a partition of $[m]$ into n subsets $S_1, \ldots, S_n$ such that $\sum_{j \in S_i} x_j = \frac{T}{n}$ for each $1 \leq i \leq n$; otherwise *No.*

Fact 4 *[GJ75,Dem14]* 3-PARTITION *is* NP-Complete.[12]

We refer to the appendix for the proof of Theorem 3, where we show that there is a polynomial time reduction from 3-PARTITION to B2LC.

5 NP-Hardness of $\texttt{REDUCIBLE}_d$

The attacks of Alwen and Blocki [AB16,AB17] exploited the fact that the Argon2i-A, Argon2i-B, iBH and Catena DAGs are not depth-robust. In general, Alwen and Blocki [AB16] showed that any (e, d)-reducible DAG G can be pebbled with cumulative cost $O\big(ne + n\sqrt{nd}\big)$. Thus, depth-robustness is a necessary condition for a secure iMHF. Recently, Alwen *et al.* [ABP17] showed that depth-robustness is sufficient for a secure iMHF. In particular, they showed that an (e, d)-depth reducible graph has $\Pi^{\parallel}_{cc}(G) \geq ed$.[13] Thus, to cryptanalyze a candidate iMHF it would be useful to have an algorithm to test for depth-robustness of an input graph G. However, we stress that (constant-factor) hardness of $\texttt{REDUCIBLE}_d$ does not directly imply that minCC is NP-Hard. To the best of our knowledge no one has explored the computational complexity of testing whether a given DAG G is (e, d)-depth robust.

[11] We may assume $\frac{T}{4n} < x_i < \frac{T}{2n}$ by taking any set of positive integers and adding a large fixed constant to all terms, as described in [Dem14].
[12] The 3PARTITION problem is called P[3,1] in [GJ75].
[13] Alwen *et al.* [ABP17] also gave tighter upper and lower bounds on $\Pi^{\parallel}_{cc}(G)$ for the Argon2i-A, iBH and Catena iMHFs. For example, $\Pi^{\parallel}_{cc}(G) = \Omega\left(n^{1.66}\right)$ and $\Pi^{\parallel}_{cc}(G) = O\left(n^{1.71}\right)$ for a random Argon2i-A DAG G (with high probability). Blocki and Zhou [BZ17] recently tightened the upper and lower bounds on Argon2i-B showing that $\Pi^{\parallel}_{cc}(G) = O\left(n^{1.767}\right)$ and $\Pi^{\parallel}_{cc}(G) = \tilde{\Omega}\left(n^{1.75}\right)$.

We have many constructions of depth-robust graphs [EGS75,PR80,Sch82,Sch83,MMV13], but the constant terms in these constructions are typically not well understood. For example, Erdös, Graham and Szemerédi [EGS75] constructed an $\big(\Omega(n), \Omega(n)\big)$-depth robust graph with $\mathsf{indeg} = O\big(\log n\big)$. Alwen *et al.* [ABP17] showed how to transform an n node (e, d)-depth robust graph with maximum indegree indeg to a $(e, d \times \mathsf{indeg})$-depth robust graph with maximum $\mathsf{indeg} = 2$ on $n \times \mathsf{indeg}$ nodes. Applying this transform to the Erdös, Graham and Szemerédi [EGS75] construction yields a constant-indegree graph on n nodes such that G is $\big(\Omega(n/\log(n)), \Omega(n)\big)$-depth robust — implying that $\Pi^{\|}_{cc}(G) = \Omega\big(\frac{n^2}{\log n}\big)$. From a theoretical standpoint, this is essentially optimal as any constant indeg DAG has $\Pi^{\|}_{cc} = O\big(\frac{n^2 \log\log n}{\log n}\big)$ [AB16]. From a practical standpoint it is important to understand the exact values of e and d for specific parameters n in each construction.

5.1 Results

We first produce a reduction from Vertex Cover which preserves approximation hardness. Let $\texttt{minREDUCIBLE}_d$ denote the problem of finding a minimum size $S \subseteq V$ such that $\mathsf{depth}(G - S) \leq d$. Our reduction shows that, for each $0 \leq d \leq n^{1-\epsilon}$, it is NP-Hard to 1.3-approximate $\texttt{minREDUCIBLE}_d$ since it is NP-Hard 1.3-approximate Vertex Cover [DS05]. Similarly, it is hard to $(2-\epsilon)$-approximate $\texttt{minREDUCIBLE}_d$ for any fixed $\epsilon > 0$ [KR08], under the Unique Games Conjecture [Kho02]. We also produce a reduction from Cubic Vertex Cover to show $\texttt{REDUCIBLE}_d$ is NP-Complete even when the input graph has bounded indegree.

The techniques we use are similar to those of Bresar et al. [BKKS11] who considered the problem of finding a minimum size d-path cover in undirected graphs (i.e., find a small set $S \subseteq V$ of nodes such that every undirected path in $G - S$ has size at most d). However, we stress that if G is a DAG, $\hat{G}$ is the corresponding undirected graph and $S \subseteq V$ is given such that $\mathsf{depth}(G - S) \leq d$ that this does not ensure that $\hat{G} - S$ contains no *undirected path* of length d. Thus, our reduction specifically address the needs for directed graphs and bounded indegree.

Theorem 5. $\texttt{REDUCIBLE}_d$ *is* NP-Complete *and it is* NP-Hard *to* 1.3*-approximate* $\texttt{minREDUCIBLE}_d$*. Under the Unique Games Conjecture, it is hard to* $(2 - \epsilon)$*-approximate* $\texttt{minREDUCIBLE}_d$*.*

Theorem 6. *Even for* $\delta = O(1)$*,* $\texttt{REDUCIBLE}_{d,\delta}$ *is* NP-Complete*.*

We defer the proofs of Theorem 5 and Theorem 6 to Appendix A. We leave open the question of efficient approximation algorithms for $\texttt{minREDUCIBLE}_d$. Lee [Lee17] recently proposed a FPT $O\left(\log\ d\right)$-approximation algorithm for the related problem d-path cover problem running in time $2^{O(d^3 \log d)} n^{O(1)}$. However, it is not clear whether the techniques could be adapted to handle directed graphs and in most interesting cryptographic applications we have $d = \Omega\left(\sqrt{n}\right)$ so the algorithm would not be tractable.

6 LP Relaxation has Large Integrality Gap

In this section, we show that the natural LP relaxation for the integer program of DAG pebbling has a large integrality gap. We show similar results for the natural LP relaxation for the integer program of $\texttt{REDUCIBLE}_d$ in Appendix B.

Let $G = (V, E)$ be a DAG with maximum indegree δ and with $V = \{1, \ldots, n\}$, where $1, 2, 3, \ldots, n$ is topological ordering of V. We start with an integer program for DAG pebbling, in Figure 6. Intuitively, $x_v^t = 1$ if we have a pebble on node v

$$\min \sum_{v \in V} \sum_{t=0}^{n^2} x_v^t. \text{ s.t.}$$

(1) $x_v^t \in \{0, 1\} \quad \forall 1 \leq v \leq n$ and $0 \leq t \leq n^2$.

(2) $x_v^0 = 0 \qquad \forall 1 \leq v \leq n.$

(3) $\forall v \in \mathsf{sinks}(G)$, $\sum_{t=0}^{n^2} x_v^t \geq 1.$

(4) $\forall v \in V \backslash \mathsf{sources}(G)$, $0 \leq t \leq n^2 - 1$

$$x_v^{t+1} \leq x_v^t + \frac{\sum_{v' \in \mathsf{parents}(v)} x_{v'}^t}{|\mathsf{parents}(v)|}.$$

Fig. 6. Integer Program for Pebbling.

during round t. Thus, Constraint 3 says that we must have a pebble on the final node at some point. Constraint 2 says that we do not start with any pebbles on G and Constraint 4 enforces the validity of the pebbling. That is, if v has parents we can only have a pebble on v in round $t+1$ if either (1) v already had a pebble during round t, or (2) all of v's parents had pebbles in round t. It is clear that the above Integer Program yields the optimal pebbling solution.

We would like to convert our Integer Program to a Linear Program. The natural relaxation is simply to allow $0 \leq x_v^t \leq 1$. However, we show that this LP has a large integrality GAP $\tilde{\Omega}\left(\frac{n}{\log n}\right)$ even for DAGs G with constant indegree. In particular, there exist DAGs with constant indegree δ for which the optimal pebbling has $\Pi_{cc}^{\parallel}(G) = \tilde{\Omega}\left(\frac{n^2}{\log n}\right)$ [ABP17], but we will provide a fractional solution to the LP relaxation with cost $O(n)$.

Theorem 7. *Let G be a DAG. Then there is a fractional solution to our LP Relaxation (of the Integer Program in Figure 6) with cost at most $4n$.*

Proof. In particular, for all time steps $t \leq n$, we set $x_i^t = \frac{1}{n}$ for all $i \leq t$, and $x_i^t = 0$ for $i > t$. For time steps $n < t \leq n + \lceil \log n \rceil$, we set $x_v^t = \min\left(1, \frac{2^{t-n}}{n}\right)$ for all $v \in V$. We first argue that this is a feasible solution. Trivially, Constraints 1 (with the LP relaxation) and 2 are satisfied. Moreover, for $t = n + \lceil \log n \rceil$, $x_v^t = 1$ for all $v \in V$, so Constraint 3 is satisfied. Note that for $1 < t \leq n$, if $x_v^t = \frac{1}{n}$, then $x_u^{t-1} = \frac{1}{n}$ for all $u < v$, so certainly $\sum_{v' \in \mathsf{parents}(v)} \frac{x_{v'}^{t-1}}{|\mathsf{parents}(v)|} \geq \frac{1}{n}$. Furthermore, $x_v^{t-1} = \min\left(1, \frac{2^{t-1-n}}{n}\right)$ for $n < t \leq n + \lceil \log n \rceil$ implies that setting

$$x_v^t = x_v^{t-1} + \sum_{v' \in \mathsf{parents}(v)} \frac{x_{v'}^t}{|\mathsf{parents}(v)|} = \frac{2^{t-1-n}}{n} + \frac{2^{t-1-n}}{n} = \frac{2^{t-n}}{n}$$

is valid. Therefore, Constraint 4 is satisfied.

Finally, we claim that $\sum_{v \in V} \sum_{t \leq n+\lceil \log n \rceil} x_v^t \leq 4n$. To see this, note that for every round $t \leq n$, we have $x_v^t \leq \frac{1}{n}$ for all $v \in V$. Thus,

$$\sum_{v \in V} \sum_{t \leq n} x_v^t \leq \sum_{v \in V} \sum_{t \leq n} \frac{1}{n} \leq \sum_{v \in V} 1 = n.$$

For time steps $n < t < n + \lceil \log n \rceil$, note that $x_v^t \leq \frac{2^{t-n}}{n}$ for all $v \in V$. Thus,

$$\sum_{v \in V} \sum_{n < t < n+\lceil \log n \rceil} x_v^t \leq \sum_{n < t < n+\lceil \log n \rceil} 2^{t-n} \leq 2n.$$

Finally, for time step $t = n + \lceil \log n \rceil$, $x_v^t = 1$ for all $v \in V$. Consequently,

$$\sum_{v \in V} \sum_{t = n+\lceil \log n \rceil} x_v^t = n.$$

Therefore,

$$\sum_{v \in V} \sum_{t \leq n+\lceil \log n \rceil} x_v^t \leq 4n.$$

One tempting way to "fix" the linear program is to require that the pebbling take at most n steps since the fractional assignment used to establish the integrality gap takes $2n$ steps. There are two issues with this approach: (1) It is not true in general that the optimal pebbling of G takes at most n steps. See Appendix C for a counter example and discussion. (2) There exists a family of DAGs with constant indegree for which we can give a fractional assignment that takes exactly n steps and costs $O(n \log n)$. Thus, the integrality gap is still $\tilde{\Omega}(n)$. Briefly, in this assignment we set $x_i^i = 1$ for $i \leq n$ and for $i < t$ we set $x_i^t = \max\left\{\frac{1}{n}, \{2^{-\mathbf{dist}(i,t+j)-j+2} : j \geq 1\}\right\}$, where $\mathbf{dist}(x, y)$ is the length of the shortest path from x to y. In particular, if $j = 1$ (we want to place a 'whole' pebble on node $j + t$ in the next round by setting $x_{j+t}^{j+t} = 1$) and node i is a parent of node $t + j$ then we have $\mathbf{dist}(i, t + j) = 1$ so we will have $x_i^{j+t} = 1$ (e.g., a whole pebble on node i during the previous round).

7 Conclusions

We initiate the study of the computational complexity of cumulative cost minimizing pebbling in the parallel black pebbling model. This problem is motivated by the urgent need to develop and analyze secure data-independent memory hard functions for password hashing. We show that it is NP-Hard to find a parallel black pebbling minimizing cumulative cost, and we provide evidence that the problem is hard to approximate. Thus, it seems unlikely that we will be able to develop tools to automate the task of a cryptanalyst to obtain strong upper/lower bounds on the security of a candidate iMHF. However, we cannot absolutely rule out the possibility that an efficient approximation algorithm exists. In fact, our results only establish worst case hardness of graph pebbling. We

cannot rule out the existance of efficient algorithms to find optimal pebblings for practical iMHF proposals such as Argon2i [BDK16] and DRSample [ABH17]. The primary remaining challenge is to either give an efficient α-approximation algorithm to find a pebbling $P \in \mathcal{P}^{\parallel}$ with $\mathsf{cc}(P) \leq \alpha \Pi^{\parallel}_{cc}(G)$ or show that $\Pi^{\parallel}_{cc}(G)$ is hard to approximate. We believe that the problem offers many interesting theoretical challenges and a solution could have very practical consequences for secure password hashing. It is our hope that this work encourages others in the TCS community to explore these questions.

Acknowledgements

We would like to thank Ioana Bercea and anonymous reviewers for helpful comments that improved the presentation of the paper. The work was supported by the National Science Foundation under NSF Awards #1649515 and #1704587. The opinions expressed in this paper are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

AB16. Joël Alwen and Jeremiah Blocki. Efficiently computing data-independent memory-hard functions. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference*, pages 241–271, 2016.

AB17. Joël Alwen and Jeremiah Blocki. Towards practical attacks on argon2i and balloon hashing. In *2017 IEEE European Symposium on Security and Privacy, EuroS&P*, pages 142–157, 2017.

ABH17. Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In *ACM CCS 17*, pages 1001–1017. ACM Press, 2017.

ABP17. Joël Alwen, Jeremiah Blocki, and Krzysztof Pietrzak. Depth-robust graphs and their cumulative memory complexity. LNCS, pages 3–32. Springer, Heidelberg, 2017.

ABW03. Martín Abadi, Michael Burrows, and Ted Wobber. Moderately hard, memory-bound functions. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*, 2003.

AS15. Joël Alwen and Vladimir Serbinenko. High Parallel Complexity Graphs and Memory-Hard Functions. In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '15, 2015. `http://eprint.iacr.org/2014/238`.

AT17. Joël Alwen and Björn Tackmann. Moderately hard functions: Definition, instantiations, and applications. In *TCC 2017, Part I*, LNCS, pages 493–526. Springer, Heidelberg, March 2017.

BCS16. Dan Boneh, Henry Corrigan-Gibbs, and Stuart E. Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In *ASIACRYPT 2016, Part I*, LNCS, pages 220–248. Springer, Heidelberg, December 2016.

BDK15. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Fast and tradeoff-resilient memory-hard functions for cryptocurrencies and password hashing. Cryptology ePrint Archive, Report 2015/430, 2015. `http://eprint.iacr.org/2015/430`.

BDK16. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2 password hash. Version 1.3, 2016. https://www.cryptolux.org/images/0/0d/Argon2.pdf.

BDKJ16. Alex Biryukov, Daniel Dinu, Dmitry Khovratovich, and Simon Josefsson. The memory-hard Argon2 password hash and proof-of-work function. Internet-Draft draft-irtf-cfrg-argon2-00, Internet Engineering Task Force, March 2016.

BK15. Alex Biryukov and Dmitry Khovratovich. Tradeoff cryptanalysis of memory-hard functions. In *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part II*, pages 633–657, 2015.

BKKS11. Bostjan Bresar, Frantisek Kardos, Ján Katrenic, and Gabriel Semanisin. Minimum k-path vertex cover. *Discrete Applied Mathematics*, 159(12):1189–1195, 2011.

BZ17. Jeremiah Blocki and Samson Zhou. On the depth-robustness and cumulative pebbling cost of Argon2i. In *TCC 2017, Part I*, LNCS, pages 445–465. Springer, Heidelberg, March 2017.

CGBS16. Henry Corrigan-Gibbs, Dan Boneh, and Stuart Schechter. Balloon hashing: Provably space-hard hash functions with data-independent access patterns. Cryptology ePrint Archive, Report 2016/027, Version: 20160601:225540, 2016. http://eprint.iacr.org/.

Cha73. Ashok K. Chandra. Efficient compilation of linear recursive programs. In *SWAT (FOCS)*, pages 16–25. IEEE Computer Society, 1973.

Coo73. Stephen A. Cook. An observation on time-storage trade off. In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, pages 29–33, New York, NY, USA, 1973. ACM.

Dem14. Erik Demaine. 6.890 algorithmic lower bounds: Fun with hardness proofs. http://ocw.mit.edu, September 2014.

DFKP15. Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 585–605. Springer, Heidelberg, August 2015.

DGN03. Cynthia Dwork, Andrew Goldberg, and Moni Naor. On memory-bound functions for fighting spam. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 426–444. Springer, 2003.

DNW05. Cynthia Dwork, Moni Naor, and Hoeteck Wee. Pebbling and proofs of work. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 37–54. Springer, Heidelberg, August 2005.

DS05. Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, pages 439–485, 2005.

EGS75. Paul Erdös, Ronald L. Graham, and Endre Szemeredi. On sparse graphs with dense long paths., 1975.

FLW13. Christian Forler, Stefan Lucks, and Jakob Wenzel. Catena: A memory-consuming password scrambler. *IACR Cryptology ePrint Archive*, 2013:525, 2013.

GJ75. Michael R Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

GLT80. John R Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The pebbling problem is complete in polynomial space. *SIAM Journal on Computing*, 9(3):513–524, 1980.

HP70. Carl E. Hewitt and Michael S. Paterson. Record of the project mac conference on concurrent systems and parallel computation, 1970.

HP10. Philipp Hertel and Toniann Pitassi. The pspace-completeness of black-white pebbling. *SIAM Journal on Computing*, 39(6):2622–2682, 2010.

Kal00. Burt Kaliski. Pkcs# 5: Password-based cryptography specification version 2.0, 2000.

Kho02. Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the Thirty-Fourth annual ACM Symposium on Theory of Computing*, pages 767–775. ACM, 2002.

KR08. Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ϵ. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

Lee17. Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1546–1558, 2017.

MMV13. Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013*, pages 373–388. ACM, January 2013.

NBF+15. Arvind Narayanan, Joseph Bonneau, Edward W Felten, Andrew Miller, and Steven Goldfeder. Bitcoin and cryptocurrency technology (manuscript), 2015. Retrieved 8/6/2015.

Per09. C. Percival. Stronger key derivation via sequential memory-hard functions. In *BSDCan 2009*, 2009.

PHC. Password hashing competition. `https://password-hashing.net/`.

PR80. Wolfgang J. Paul and Rüdiger Reischuk. On alternation II. A graph theoretic approach to determinism versus nondeterminism. *Acta Inf.*, 14:391–403, 1980.

RD16. Ling Ren and Srinivas Devadas. Proof of space from stacked expanders. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Proceedings, Part I*, pages 262–285, 2016.

RD17. Ling Ren and Srinivas Devadas. Bandwidth hard functions for ASIC resistance. In *TCC 2017, Part I*, LNCS, pages 466–492. Springer, Heidelberg, March 2017.

Sch82. Georg Schnitger. A family of graphs with expensive depth reduction. *Theor. Comput. Sci.*, 18:89–93, 1982.

Sch83. Georg Schnitger. On depth-reduction and grates. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 323–328. IEEE Computer Society, 1983.

SS78. John E. Savage and Sowmitri Swamy. Space-time trade-offs on the fft algorithm. *IEEE Transactions on Information Theory*, 24(5):563–568, 1978.

SS79a. John E. Savage and Sowmitri Swamy. Space-time tradeoffs for oblivious interger multiplications. In Hermann A. Maurer, editor, *ICALP*, volume 71 of *Lecture Notes in Computer Science*, pages 498–504. Springer, 1979.

SS79b. Sowmitri Swamy and John E. Savage. Space-time tradeoffs for linear recursion. In Alfred V. Aho, Stephen N. Zilles, and Barry K. Rosen, editors, *POPL*, pages 135–142. ACM Press, 1979.

Tom78. Martin Tompa. Time-space tradeoffs for computing functions, using connectivity properties of their circuits. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 196–204, New York, NY, USA, 1978. ACM.

A Missing Proofs

Reminder of Lemma 1. *If the* B2LC *instance has a valid solution, then* $\Pi^{\parallel}_{cc}(G_{\texttt{B2LC}}) \leq \tau cmn + 2cmn + 2ckm + 1$.

Proof of Lemma 1. Suppose the given instance of B2LC has a valid solution, $\{x_{i,j}\}$. Recall that a pebble must pass m times through each of the τ chains of length c representing each variable. For a set k, let $x_{k,j_1} \leq x_{k,j_2} \leq \ldots \leq x_{k,j_n}$. We start the k^{th} pass through the τ chains by placing a pebble on $C^1_{j_n}, \ldots, C^\tau_{j_n}$, the τ chains representing variable j_n. At each subsequent time step, we move the existing pebbles to the next node along the chain. When the pebbles on chains $C^1_{j_n}, \ldots, C^\tau_{j_n}$ reach the $(x_{k,j_n} - x_{k,j_{n-1}} + 1)^{th}$ nodes, we place a pebble on $C^1_{j_{n-1}}, \ldots, C^\tau_{j_{n-1}}$, the τ chains representing variable j_{n-1}. We continue this process by moving existing pebbles to the next node along each chain for each subsequent time step.

When pebbles on chains $C^1_{j_l}, \ldots, C^\tau_{j_l}$ reach the $(x_{k,j_l} - x_{k,j_{l-1}} + 1)^{th}$ nodes, we place a pebble on $C^1_{j_{l-1}}, \ldots, C^\tau_{j_{l-1}}$, the τ chains representing variable j_{l-1}. Thus, the gadgets $E_1, \ldots, E_m$ which are satisfied by $x_{k,1}, \ldots, x_{k,m+1}$ can be pebbled during this pass, since by construction, we offset the positions of the pebbles by the appropriate distances. When a pebble reaches the end of its chain, we remove the pebble. Hence, we see that each chain has c time steps with pebbles, and each of those steps needs only one pebble. There are n variables, τ chains representing each variable, and m passes through each chain. Across all variable chains, the cumulative complexity is τcmn since there are n variables and τ chains representing each variable.

Similarly, making m walks through the paths $C^1_j, \ldots, C^\tau_j$ allows us to pebble the path M_j ($j \leq n$). These paths each have length cm and we keep at most one pebble on M_j at any point in time. There may be a delay of up to c time steps between consecutive walks through the paths $C^1_j, \ldots, C^\tau_j$ during which we cannot progress our pebble through the path M_j. However, there are at most $2cm$ total steps in which we have a pebble on M_j. Thus, the cumulative complexity across all paths $M_1, ..., M_n$ is at most $2cmn$.

Likewise, since each equation gadget E_j is represented by a chain, one pebble at each time step suffices for each of these paths. We may have to keep a pebble on the gadget E_j while we make m walks through the paths, but we keep this pebble on E_j for at most $2cm$ steps (each walk takes c steps and the delay between consecutive walks is at most c steps). Since there are k equations, and there is a chain for each equation, the cumulative complexity across all equation chains is at most $2ckm$.

There is one final sink node, so the cumulative complexity for an instance of B2LC with a valid solution is at most $\tau cmn + 2cmn + 2ckm + 1$. □

Claim 8 will be useful in our proof of Lemma 2.

Claim 8 *Any pebbling strategy $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\parallel}_{G_{\text{B2LC}}}$ with $\Pi^{\parallel}_{cc}(P) = \Pi^{\parallel}_{cc}(G)$ must satisfy the following property: for all $i \in [n], j \in [\tau]$ and all pebbling rounds $y \in [t]$ and $z \in [c]$ we have $v_i^{j,z} \in P_y \leftrightarrow \{v_i^{j',z} : j' \in [\tau]\} \subseteq P_y$. In particular, whenever we have a pebble on node $v_i^{j,z}$ (the z'th node in the j'th path gadget C_i^j for variable x_i) we also have pebbles on each of the nodes $v_i^{1,z}, \ldots, v_i^{\tau,z}$.*

Proof of Claim 8. Let $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\parallel}_{G_{\text{B2LC}}}$ be a pebbling that does not satisfy our property. We will construct another legal pebbling $P' = (P'_0, \ldots, P'_t) \in \mathcal{P}^{\parallel}(G_{\text{B2LC}})$ with $\Pi^{\parallel}_{cc}(G) \leq \mathsf{cc}(P') < \mathsf{cc}(P)$. For time step y, set

$$P'_y = P_y \setminus UNSYNC_y$$

where

$$UNSYNC_y = \left\{ v_i^{j,z} : i \in [n], j \in [\tau], z \in [c] \text{ and } \{v_i^{j',z} : j' \in [\tau]\} \not\subset P_y \right\} .$$

We clearly have $\left|P'_y\right| \leq |P_y|$ at each time step y. Furthermore, because P does not satisfy our property we must have $|P_y| > \left|P'_y\right|$ for some step y. Thus,

$$\mathsf{cc}(P') = \sum |P'_y| < \sum |P_y| = \mathsf{cc}(P) .$$

It remains to show that P' is a legal pebbling i.e., $\forall y \forall v \in P'_{y+1}$ we have $\mathsf{parents}(P'_{y+1}) \subseteq P'_y$. For a node $v \in P'_{y+1}$, we have four cases:

1. For some variable x_i the node v is a part of one of the τ gadgets for that variable i.e., $v = v_i^{j,z} \in C_i^j$ for some $j \in [\tau]$. By construction of P' we must also have $v_i^{j',z} \in P'_{y+1}$ for each $j' \in [\tau]$. Since $v_i^{j',z} \in P'_{y+1} \subseteq P_{y+1}$, we must have $\mathsf{parents}(v_i^{j',z}) = \{v_i^{j',z-1}\} \subset P_y$ by the legality of the original pebbling P. Thus, $\{v_i^{j',z-1} : j' \in [\tau]\} \subseteq P'_y$ since $UNSYNC_y \cap \{v_i^{j',z-1} : j' \in [\tau]\} = \emptyset$. It follows that $\mathsf{parents}(v) \subseteq P'_y$.
2. $v = z_i^{p+qc} \in M_i$. In this case we observe that, since $v \in P_{y+1}$, we have $\mathsf{parents}(v) \subseteq P_y$ by the legality of the original pebbling P. By construction, of G_{B2LC} we have $\mathsf{parents}(z_i^{p+qc}) = \{v_i^{j',p} : j \in [\tau]\} \cup \{z_i^{p+qc-1}\}$. In the construction of P'_y we would not discard any of these pebbles since $\mathsf{parents}(z_i^{p+qc}) \cap UNSYNC_y = \emptyset$. Thus, we have $\mathsf{parents}(v) \subseteq P'_y$.
3. $v = e_i^k \in E_i$ for some equation gadget E_i. The proof that $\mathsf{parents}(v) \subseteq P'_y$ is essentially the same as in case 2.
4. v is a sink. By legality of P we have $\mathsf{parents}(v) \subseteq P_y$. In this case we note that parents(sink) is disjoint from all of the variable gadgets C_i^j. Since, $P'_y = P_Y \setminus UNSYNC_y$ can only remove pebbles on the variable gadgets C_i^j it follows that $\mathsf{parents}(v) \subseteq P'_y$.

In each case, $\mathsf{parents}(v) \subseteq P'_y$ so P' is a legal pebbling. □

Reminder of Lemma 2. *If the* B2LC *instance does not have a valid solution, then* $\Pi^{\parallel}_{cc}(G_{\texttt{B2LC}}) \geq \tau cmn + \tau$.

Proof of Lemma 2. Suppose the given instance of B2LC does not have a valid solution and let $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\parallel}_{G_{\texttt{B2LC}}}$ be given such that $\mathsf{cc}(P) = \Pi^{\parallel}_{cc}(G_{\texttt{B2LC}})$ i.e. P is optimal. We first observe that in any legal pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\parallel}(G_{\texttt{B2LC}})$ we must walk a pebble down each of the paths $M_1, \ldots, M_n$ of length cm. Let t_i^z denote the first time step in which we place a pebble on v_i^z — the z'th node on path M_i. Clearly, $t_i^z < t_i^{z+1}$ for $z < cm$ and at time $t_i^z - 1$ we must have a pebble on all nodes in $\mathsf{parents}(v_i^z)$. In particular, $v_i^{1,y}, \ldots, v_i^{\tau,y} \in P_{t_i^z - 1}$ where $y = z \pmod c$. Thus, for each node $v_i^{j,z}$ with $i \in [n], j \in [\tau], z \in [c]$ there are at least m distinct rounds $y \in \left\{t_i^z, t_i^{z+c}, \ldots, t_i^{z+c(m-1)}\right\}$ during which $v_i^{j,z} \in P_y$.

We say that the pebbling P "cheats" if it does not correspond to a valid B2LC solution. Formally, P is a "cheating" pebbling if for some node $v_i^{j,z}$ there are at least $m+1$ distinct rounds $y \in \{y_1, \ldots, y_{m+1}\}$ during which $v_i^{j,z} \in P_y$. By Claim Claim 8 we must have $\left\{v_i^{1,z}, \ldots, v_i^{\tau,z}\right\} \in P_y$ for each $y \in \{y_1, \ldots, y_{m+1}\}$. Thus, if P is a cheating pebbling we have

$$\begin{aligned}
\mathsf{cc}(P) &\geq \sum_{y \in [t]} \sum_{j \in [\tau]} \left| P_y \cap v_i^{j,z} \right| + \sum_{y \in [t]} \sum_{j \in [\tau]} \sum_{\substack{i \in [n], z \in [c] \ s.t \\ (i',z') \neq (i,z)}} \left| P_y \cap v_i^{j,z} \right| \\
&\geq \tau(m+1) + \sum_{j \in [\tau]} \sum_{\substack{i \in [n], z \in [c] \ s.t \\ (i',z') \neq (i,z)}} m \\
&= \tau(m+1) + (cn\tau m - \tau m) \\
&= \tau cmn + \tau \ .
\end{aligned}$$

Any non-cheating pebbling corresponds to a valid B2LC solution. If an equation $x_{\alpha_i} + c_i + x_{\beta_i}$ is not satisfied by one of the assignments in the B2LC solution then there is no legal way to pebble the equation chain E_i without cheating because at no point during the m walks are both variables in the equation offset by the correct amount. Thus, any instance of B2LC which does not have a valid solution requires that $\Pi^{\parallel}_{cc}(G_{\texttt{B2LC}}) \geq \tau cmn + \tau$. □

Reminder of Theorem 2. $\texttt{minCC}_\delta$ *is* NP-Complete *for each* $\delta \geq 2$.

Proof of Theorem 2. (sketch) We sketch the construction for $\delta = 2$ due to its similarity to the general case where the indegree is not restricted and we highlight the differences between the constructions. Although we maintain τ chains representing each variable, we can no longer maintain the gadgets for the equation, $E_1, \ldots, E_m$, for which each vertex has indegree $2\tau + 1$, one edge from its predecessor in the gadget, and 2τ edges from the chains representing the variables involved in the equation. Instead, in place of the 2τ edges, we construct a binary tree, where the bottom layer of the tree has at least $\binom{\tau}{2}$ nodes. Each of the $\binom{\tau}{2}$ connects to a separate instance of the chains representing the variables

involved in the equation. Thus, if the equation involves variables x_i and x_j, then each of the $\binom{\tau}{2}$ nodes will have an edge from one of the τ chains representing x_i, and an edge from one of the τ chains representing x_j, offset by an appropriate amount. This construction ensures that the root of the tree is pebbled only if all equations are satisfied, but any "dishonest" walk along the chains will require at least τ additional pebbles. Similarly, we replace the 2τ edges in the paths $P_1, \ldots, P_n$ of length cm with binary trees with base $\binom{\tau}{2}$. Finally, we replace the $m+n$ incident edges to the sink node with a binary tree with base $\binom{m+n}{2}$. Since each of these terms are polynomial in c, m, n, there exists a τ that is also polynomial in c, m, n which allows us to distinguish between honest walks and dishonest walks. As a result, we can decide between instances of B2LC. □

Reminder of Theorem 3. B2LC *is* NP-Complete.

Proof of Theorem 3. Given an instance S of 3-PARTITION, first sort S so that $S = \{s_1, s_2, \ldots, s_m\}$ and $s_i \leq s_j$ for any $i \leq j$. Let $T = \sum_{i=1}^{m} s_m$. We then create $mn = 3n^2$ equations:

$$\begin{gathered}
x_1 + s_1 = x_2, \quad x_2 + s_2 = x_3, \quad \ldots, \quad x_m + s_m = x_{m+1}, \\
x_1 + 0 = x_2, \quad x_2 + 0 = x_3, \quad \ldots, \quad x_m + 0 = x_{m+1}, \\
x_1 + T = x_2, \quad x_2 + T = x_3, \quad \ldots, \quad x_m + T = x_{m+1}, \\
x_1 + 2T = x_2, \quad x_2 + 2T = x_3, \quad \ldots, \quad x_m + 2T = x_{m+1}, \\
x_1 + (n-2)T = x_2, \quad x_2 + (n-2)T = x_3, \quad \ldots, \quad x_m + (n-2)T = x_{m+1},
\end{gathered}$$

Finally, we create the additional n equations:

$$x_1 + \frac{T}{n} + 3(i-1)(n-2)T = x_{m+1}, \tag{1}$$

for $i \in [n]$. This gives a total of $3n^2 + n$ equations so that the reduction is clearly achieved in polynomial time.

Recall that B2LC is true if and only if there exist $\{a_{i,j}\}$, $i \in [n]$, $j \in [m]$ so that each equation $x_i + c_{i,j} = x_j$ is satisfied by the assigning $x_i = a_{i,k}$ and $x_j = a_{j,k}$ for some k. We show that there exists a solution to 3-PARTITION if and only if there exists a solution to B2LC.

Suppose there exists a partition of S into n triplets $S_1, S_2, \ldots, S_n$ so that the sum of the integers in each triplet is the same, and equals $\frac{T}{n}$. We set $a_{1,1} = 0$ and for each s_i that appears in S_1, we choose to satisfy the equation, $a_{1,i} + s_i = a_{1,i+1}$. Otherwise, if s_i does not appear in S_1, we choose to satisfy the equation $a_{1,i} + 0 = a_{1,i+1}$.

Suppose that $a_{i,j}$ are defined for all $i < k$. Then we set $a_{k,1} = 0$ and for each s_i that appears in S_k, we choose to satisfy the equation, $a_{k,i} + s_i = a_{k,i+1}$. If s_i appears in S_i for some $i < k$, then we choose to satisfy the equation $a_{k,i} + (i-2)T = a_{k,i+1}$. Otherwise, if s_i does not appear in $S_1, \ldots, S_k$, we choose to satisfy the equation $a_{k,i} + (i-1)T = a_{k,i+1}$.

Thus, to get $a_{i,m+1}$ from $a_{i,1}$, we add in three elements whose sum is $\frac{T}{n}$. That is, we pick three unused elements of S, say s_t, s_u, s_v, and we choose to satisfy

the equations $x_t + s_t = x_{t+1}$, $x_u + s_u = x_{u+1}$, and $x_v + s_v = x_{v+1}$. We then add in $3(i-1)$ instances of $(i-2)T$, for each of the elements which appear in $S_1, \ldots, S_{i-1}$. Finally, we add in $(3n-3i)$ instances of $(i-1)T$. Hence, it follows that

$$\begin{aligned} a_{i,1} + \frac{T}{n} + 3(i-1)(i-2)T + (3n-3i)(i-1)T &= a_{i,m+1} \\ a_{i,1} + \frac{T}{n} + (3in - 3n - 6i + 6)T &= a_{i,m+1} \\ a_{i,1} + \frac{T}{n} + 3(i-1)(n-2)T &= a_{i,m+1}, \end{aligned}$$

so that all equations in 1 are satisfied. Thus, a solution for 3-PARTITION yields a solution for B2LC.

Suppose there exists a solution for the above instance of B2LC. Observe that since there are n instances of the equation $x_1 + \frac{T}{n} + 3(i-1)(n-2)T = x_{m+1}$, then the equation must hold for each $a_{i,1}, a_{i,m+1}$. Thus, $a_{i,1} + \frac{T}{n} \equiv a_{i,m+1} \pmod{T}$ for all i. Hence, to obtain $a_{1,m+1}$ from $a_{1,1}$, we must take three equations of the form $x_i + s_i = x_{i+1}$ since the summing less than three elements in S is less than $\frac{T}{n}$, while the summing more than three elements in S is more than $\frac{T}{n}$ but less than $T + \frac{T}{n}$ (as each element of S is greater than $\frac{T}{4n}$ and less than $\frac{T}{2n}$ and the sum of all elements in S is T). Say that these three equations use the terms $s_{i_1}, s_{i_2}, s_{i_3}$. Then we let $S_1 = \{s_{i_1}, s_{i_2}, s_{i_3}\}$, which indeed sums to $\frac{T}{n}$.

Similarly, to obtain $a_{k,m+1}$ from $a_{k,1}$, we must take three equations of the form $x_i + s_i = x_{i+1}$. Since there are $3n$ equations of this form which must be satisfied and each of the n assignments of the form $a_{i,1}, \ldots, a_{i,m+1}$ (where $1 \leq i \leq n$) uses exactly three of these equations, then each assignment of $a_{i,1}, \ldots, a_{i,m+1}$ must satisfy a disjoint triplet of the $3n$ equations. Say that the three satisfied equations for $a_{i,1}, \ldots, a_{i,m+1}$ are $s_{k_1}, s_{k_2}, s_{k_3}$. Then we let $S_k = \{s_{k_1}, s_{k_2}, s_{k_3}\}$, which indeed sums to $\frac{T}{n}$.

Therefore, we have a partition of S into triplets, which each sum to $\frac{T}{n}$, as desired. Thus, a solution for B2LC yields a solution for 3-PARTITION. □

Reminder of Theorem 5. $\texttt{REDUCIBLE}_d$ *is* `NP-Complete` *and it is* `NP-Hard` *to* 1.3*-approximate* $\texttt{minREDUCIBLE}_d$*. Under the Unique Games Conjecture, it is hard to* $(2-\epsilon)$*-approximate* $\texttt{minREDUCIBLE}_d$*.*

Proof of Theorem 5. We provide a reduction from `VC` to $\texttt{REDUCIBLE}_d$. Given an instance $G = (V, E)$ of `VC`, arbitrarily label the vertices $1, \ldots, n$, where $n = |V|$, and direct each edge of E so that $1, \ldots, n$ is a topological ordering of the nodes. For each node i we add two directed paths (where path length is the number of edges in the path): (1) a path of length $i-1$ with an edge from the last node on the path to node i, (2) a path of length $n-i$ with an edge from node i to the first node of the path. To avoid abuse of notation, let U represent the original n vertices (from the given instance of `VC`) in the modified graph.

We claim `VC` has a vertex cover of size at most k if and only if the resulting graph is (k, n)-reducible. Indeed, if there exists a vertex cover of size k, we remove

the corresponding k vertices in the resulting construction. Then there are no edges connecting vertices of U. By construction, any path of length $n-1$ must contain at least two vertices of U. Hence, the resulting graph is (k,n)-reducible.

On the other hand, suppose that there is no vertex cover of size k. Given a set S of $k = |S|$ vertices we say that a node $u \in U$ is "untouched" by S if $u \notin S$ and S does not contain any vertex from the chain(s) we connected to node u. If there is no vertex cover of size k, then removing any set S of $k = |S|$ vertices from the graph leaves an edge (u, v) with the following properties: (1) $u, v \in U$, (2) u and v are both untouched by S. Suppose u has label i. Then the (untouched) directed path which ends at u has length $i-1$. Similarly, there still exists some directed path of length $\geq n-i$ which begins at v. Thus, there exists a path of length at least n, so the resulting graph is not (k,n)-reducible. □

Reminder of Theorem 7. *Let G be a DAG. Then there is a fractional solution to our LP Relaxation (of the Integer Program in Figure 6) with cost at most $4n$.*
Proof of Theorem 7. In particular, for all time steps $t \leq n$, we set $x_i^t = \frac{1}{n}$ for all $i \leq t$, and $x_i^t = 0$ for $i > t$. For time steps $n < t \leq n + \lceil \log n \rceil$, we set $x_v^t = \min\left(1, \frac{2^{t-n}}{n}\right)$ for all $v \in V$. We first argue that this is a feasible solution. Trivially, Constraints 1 (with the LP relaxation) and 2 are satisfied. Moreover, for $t = n + \lceil \log n \rceil$, $x_v^t = 1$ for all $v \in V$, so Constraint 3 is satisfied. Note that for $1 < t \leq n$, if $x_v^t = \frac{1}{n}$, then $x_u^{t-1} = \frac{1}{n}$ for all $u < v$, so certainly $\sum_{v' \in \mathsf{parents}(v)} \frac{x_{v'}^{t-1}}{|\mathsf{parents}(v)|} \geq \frac{1}{n}$. Furthermore, $x_v^{t-1} = \min\left(1, \frac{2^{t-1-n}}{n}\right)$ for $n < t \leq n + \lceil \log n \rceil$ implies that setting

$$x_v^t = x_v^{t-1} + \sum_{v' \in \mathsf{parents}(v)} \frac{x_{v'}^t}{|\mathsf{parents}(v)|} = \frac{2^{t-1-n}}{n} + \frac{2^{t-1-n}}{n} = \frac{2^{t-n}}{n}$$

is valid. Therefore, Constraint 4 is satisfied.

Finally, we claim that $\sum_{v \in V} \sum_{t \leq n + \lceil \log n \rceil} x_v^t \leq 4n$. To see this, note that for every round $t \leq n$, we have $x_v^t \leq \frac{1}{n}$ for all $v \in V$. Thus,

$$\sum_{v \in V} \sum_{t \leq n} x_v^t \leq \sum_{v \in V} \sum_{t \leq n} \frac{1}{n} \leq \sum_{v \in V} 1 = n.$$

For time steps $n < t < n + \lceil \log n \rceil$, note that $x_v^t \leq \frac{2^{t-n}}{n}$ for all $v \in V$. Thus,

$$\sum_{v \in V} \sum_{n < t < n + \lceil \log n \rceil} x_v^t \leq \sum_{n < t < n + \lceil \log n \rceil} 2^{t-n} \leq 2n.$$

Finally, for time step $t = n + \lceil \log n \rceil$, $x_v^t = 1$ for all $v \in V$. Consequently,

$$\sum_{v \in V} \sum_{t = n + \lceil \log n \rceil} x_v^t = n.$$

Therefore,

$$\sum_{v \in V} \sum_{t \leq n + \lceil \log n \rceil} x_v^t \leq 4n.$$

□

B Integrality Gaps for $\texttt{REDUCIBLE}_d$

We now suggest a natural integer program for $\texttt{REDUCIBLE}_d$ and show that the integrality gap is quite large $\tilde{\Omega}(n)$.

$$\begin{array}{ll} \min \sum_{v \in V} x_v. \text{ s.t.} & (2)\ 0 \leq d_{u,v} \leq d \quad \forall (u,v) \in V^2 \\ (1)\ x_v \in \{0,1\} \quad \forall 1 \leq v \leq n. & (3)\ d_{w,v} \geq d_{w,u} + 1 - (d+1)(x_u + x_v) \quad \forall w \in V, (u,v) \in E \end{array}$$

Fig. 7. Integer Program for $\texttt{REDUCIBLE}_d$.

Intuitively, setting $x_v = 1$ means that we include $v \in S$ and $d_{u,v}$ represents the length of the maximum length directed path from u to v in the graph $G - S$. Constraint 2 requires that the longest path has length at most d, and Constraint 3 ensures that $d_{w,v}$ upper bounds the length of the longest path from w to v in $G - S$. If we have a path from w to u of length k in $G - S$ and $u, v \notin S$ then there is a path of length $k + 1$ from w to v in $G - S$ — if $u \in S$ or $v \in S$ then the $-n(x_u + x_v)$ term effectively eliminates the constraint since this particular path does not exist in $G - S$.

The LP relaxation is obtained by changing Constraint 1 to $0 \leq x_v \leq 1$. To see that the LP relaxation has high integrality gap we first observe that there is a family of $(\Omega(n), \Omega(n))$-depth robust DAGs G_n with $\mathsf{indeg}(G_n) \leq \log n$. By Theorem 9 the LP relaxation for G_n has a solution with cost at most $n/d = \theta(1)$, but the Integer Program must have cost at least $\Omega(n)$. Thus, the integrality gap is $\Omega(n)$. Even if we require $\mathsf{indeg}(G_n) = 2$ then we still have a family of $(\Omega(n/\log n), \Omega(n))$-depth robust DAGs [ABP17] so we get an integrality gap of $\Omega(n/\log n)$.

Theorem 9. *For* any *DAG G the LP relaxation (of the Integer Program in Figure 7) has a solution with cost at most n/d.*

Proof. Set $x_v = \frac{1}{d}$ for all $v \in V$ and set $d_{u,v} = 0$ for all $u, v \in V^2$.

C On Pebbling Time and Cumulative Cost

In this section we address the following question: is is necessarily case that an optimal pebbling of G (minimizing cumulative cost) takes n steps? For example, the pebbling attacks of Alwen and Blocki [AB16,AB17] on depth-reducible graphs such as Argon2i-A, Argon2i-B, Catena all take *exactly* n steps. Thus, it seems natural to conjecture that the optimal pebbling of G *always* finishes in $\mathsf{depth}(G)$ steps. In this section we provide a concrete example of an DAG G (with n nodes and $\mathsf{depth}(G) = n$) with the property that *any* optimal pebbling of G *must* take more than n steps. More formally, let $\Pi^{\|}(G, t)$ denote the set of all legal pebblings of G that take at most t pebbling rounds. We prove that $\min_{P \in \Pi^{\|}(G,t)} \mathsf{cc}(P) > \min_{P \in \Pi^{\|}(G)} \mathsf{cc}(P) = \Pi^{\|}_{cc}(G)$.

Theorem 10. *There exists a graph G with $n = 16$ nodes and $\mathsf{depth}(G) = n$ s.t. for some $t > 0$, $\frac{\min_{P \in \Pi^{\parallel}(G,t)} \mathsf{cc}(P)}{\Pi^{\parallel}_{cc}(G)} \geq \frac{28}{27}$.*

Proof. Consider the following DAG G on 16 nodes $\{1, \ldots, 16\}$ with the following directed edges (1) $(i, i+1)$ for $1 \leq i < 16$, (2) $(i, i+9)$ for $1 \leq i \leq 5$ and (3) $(i, i+7)$ for $8 \leq i \leq 9$. We first show in Claim C that there is a pebbling P with cost 27 that takes 18 rounds. Theorem 10 then follows from Claim C where we show that any $P \in \Pi^{\parallel}(G, 16)$ has cumulative cost at least 28. Intuitively, any legal pebbling must either delay before pebbling nodes 15 and 16 or during rounds $15 - i$ (for $0 \leq i \leq 5$) we *must* have at least one pebble on some nodes in the set $\{9, 8, ..., 9 - i\}$.

Claim. For the above DAG G we have $\Pi^{\parallel}_{cc}(G) \leq 27$.

Proof. Consider the following pebbling: $P_0 = \emptyset, P_1 = \{1\}, P_2 = \{2\}, P_3 = \{3\}, P_4 = \{4\}, P_5 = \{5\}, P_6 = \{6\}, P_7 = \{7\}, P_{\{}8\} = \{8\}, P_9 = \{1, 9\}, P_{10} = \{2, 10\}, P_{11} = \{3, 11\}, P_{12} = \{4, 12\}, P_{13} = \{5, 13\}, P_{14} = \{6, 14\}, P_{15} = \{7, 14\}, P_{16} = \{8, 14\}, P_{17} = \{9, 15\}, P_{18} = \{16\}$. It is easy to verify that $\mathsf{cc}(P) = 9 + 2 * 9 = 27$ since there are 9 steps in which we have one pebble on G and 9 steps in which we have two pebbles on G.

Claim. For the above DAG G we have $\min_{P \in \Pi^{\parallel}(G,16)} \mathsf{cc}(P) \geq 28$.

Proof. Let $P = (P_1, \ldots, P_{16}) \in \Pi^{\parallel}(G, 16)$ be given. Clearly, $i \in P_i$ for each round $1 \leq i \leq 16$. To pebble nodes 15 and 16 on steps 15 and 16 we must have $9 \in P_{15}$ and $8 \in P_{14}$. By induction, this means that $P_{14-i} \cap \{8, ..., 8 - i\} \neq \emptyset$ for each $i > 0$. To pebble node $9 + i$ at time $9 + i$ we require that $i \in P_{8+i}$ for each $1 \leq i \leq 5$. These observations imply that $|P_9| \geq 3$, $|P_{10}| \geq 3$, ..., $|P_{13}| \geq 3$. We also have $|P_{15}| \geq 2$ and $|P_{14}| \geq 2$. We also have $|P_i| \geq 1$ for all $1 \leq i \leq 16$. The cost of rounds 1–8 and round 16 is at least 9. The cost of rounds 9–13 is at least 15 and the cost of rounds 14 and 15 is at least 4. Thus, $\mathsf{cc}(P) \geq 28$.

Open Questions: Theorem 10 shows that $\Pi^{\parallel}_{cc}(G)$ can be smaller than $\min_{P \in \Pi^{\parallel}(G,t)} \mathsf{cc}(P)$, but how large can this gap be in general? Can we prove upper/lower bounds on the ratio: $\frac{\min_{P \in \Pi^{\parallel}(G,t)} \mathsf{cc}(P)}{\Pi^{\parallel}_{cc}(G)}$ for any n node DAG G? Is it true that $\frac{\min_{P \in \Pi^{\parallel}(G,t)} \mathsf{cc}(P)}{\Pi^{\parallel}_{cc}(G)} \leq c$ for some constant c? If not does this hold for n node DAGs G with constant indegree? Is it true that the optimal pebbling of G always takes at most cn steps for some constant c?

D NP-Hardness of `minST`

Recall that the space-time complexity of a graph pebbling is defined as $\mathsf{st}(P) = t \times \max_{1 \leq i \leq t} |P_i|$. We define `minST` and `minSST` based on whether the graph pebbling is parallel or sequential. Formally, the decision problem `minST` is defined as follows:

Input: a DAG G on n nodes and an integer $k < n(n+1)/2$.
Output: *Yes*, if $\min_{P \in \mathcal{P}_G^{\parallel}} \mathtt{st}(P) \leq k$; otherwise *No*.

The decision problem `minSST` is defined as follows:
Input: a DAG G on n nodes and an integer $k < n(n+1)/2$.
Output: *Yes*, if $\Pi_{st}(G) \leq k$; otherwise *No*.

Gilbert *et al.*[GLT80] provide a construction from any instance of TQBF to a DAG G_{TQBF} with pebbling number $3n+3$ if and only if the instance is satisfiable. Here, the pebbling number of a DAG G is $\min_{P=(P_1,\ldots,P_t)\in\mathcal{P}^{\parallel}} \max_{i\leq t} |P_i|$ is the number of pebbles necessary to pebble G. An important gadget in their construction is the so-called pyramid DAG. We use a triangle with the number k inside to denote a k-pyramid (see Figure 8 for an example of a 3-pyramid). The key property of these DAGs is that *any* legal pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\parallel}(Pyramid_k)$ of a k-pyramid requires at least $\min_i |P_i| \geq k$ pebbles on the DAG at some point in time. Another gadget, which appears in Figure 9, is the existential quantifier gadget, which requires that s_i, $s_i - 1$, and $s_i - 2$ pebbles must be placed in each of the pyramids to ultimately pebble q_i.

Remark: We note that [GLT80] focused on sequential pebblings ($P \in \mathcal{P}$) in their analysis, but their analysis extends to parallel pebblings ($P \in \mathcal{P}^{\parallel}$) as well.

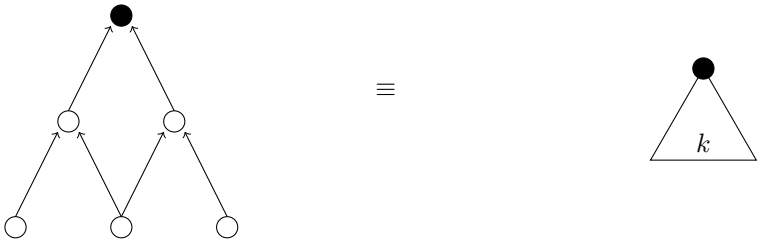

Fig. 8. A 3-Pyramid.

We observe that any instance of TQBF where each quantifier is an existential quantifier requires at most a quadratic number of pebbling moves. Specifically, we look at instances of `3-SAT`, such as in Figure 10. In such a graph representing an instance of `3-SAT`, the sink node to be pebbled is q_n. By design of the construction, any true statement requires exactly three pebbles for each pyramid representing a clause. On the other hand, a false clause requires four pebbles, so that false statements require more pebbles. Thus, by providing extraneous additions to the construction which force the number of pebbling moves to be a known constant, we can extract the pebbling number, given the space-time complexity. For more details, see the full description in [GLT80].

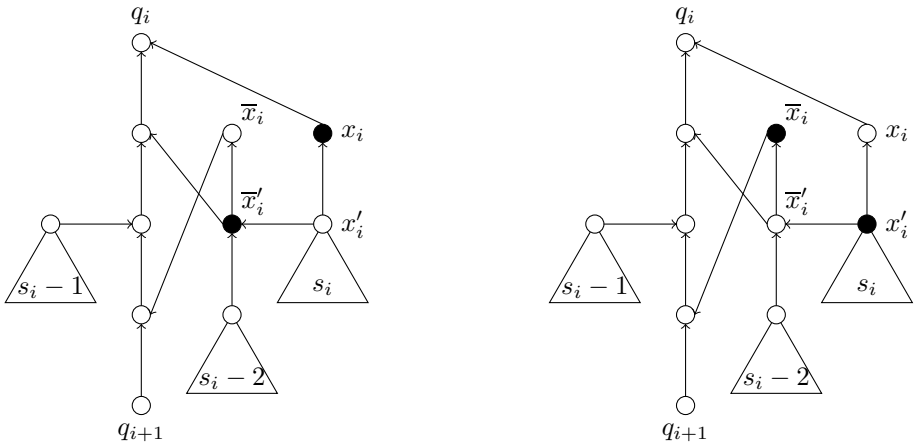

Fig. 9. An existential quantifier, with x_i set to true in the left figure and x_i set to false in the right figure.

Lemma 3. *[GLT80] The quantified Boolean formula $Q_1x_1Q_2x_2\cdots Q_nx_nF_n$ is true if and only if the corresponding DAG G_{TQBF} has pebbling number $3n+3$.*

Lemma 4. *Suppose that we have a satisfiable TQBF formula $Q_1x_1Q_2x_2\cdots Q_nx_nF_n$ with $Q_i = \exists$ for all $i \leq n$. Then there is a legal sequential pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}(G_{TQBF})$ of the corresponding DAG G_{TQBF} from [GLT80] with $t \leq 6n^2 + 33n$ pebbling moves and $\max_{i \leq t} |P_i| \leq 3n + 3$.*

Proof. We describe the pebbling strategy of Gilbert *et al.* [GLT80], and analyze the pebbling time of their strategy. Let $T(i)$ be the time it takes to pebble q_i in the proposed construction for any instance with i variables, i clauses, and only existential quantifiers.

Suppose that x_i is allowed to be true for the existential quantifier $Q_i = \exists$. Then vertex x'_i is pebbled using s_i moves, where $s_i = 3n - 3i + 6$. Similarly, vertices d_i and f_i are pebbled using $s_i - 1$ and $s_i - 2$ moves respectively. Additionally, f_i is moved to $\overline{x}'_i$ and then x_i is moved to x'_i in the following step, for a total of two more moves. We then pebble q_{i+1} using $T(i+1)$ moves and finish by placing a pebble on $\overline{x}_i$ and moving it to c_i, b_i, a_i, and q_i, for five more moves. Finally, we use six more moves to pebble an additional clause. Thus, in this case,

$$T_{true}(i) = s_i + (s_i - 1) + (s_i - 2) + 13 + T(i+1).$$

On the other hand, if x_i is allowed to be false for the existential quantifier $Q_i = \exists$, then first we pebble x'_i, d_i, and f_i sequentially, using s_i, $s_i - 1$, and $s_i - 2$ moves respectively. We then move the pebble from f_i to $\overline{x}'_i$ and then to $\overline{x}_i$, for a total of two more moves. We then pebble q_{i+1} using $T(i+1)$ moves. The pebble on q_{i+1} is subsequently moved to c_i and then b_i, using two more moves. Picking up all pebbles except those on b_i and x'_i, and using them to pebble f_i takes $s_i - 2$

more moves. Additionally, the pebble on f_i is moved to $\overline{x}'_i$ and then a_i, while the pebble on x'_i is moved to x_i and then q_i, for four more moves. Finally, we use six more moves to pebble an additional clause. In total,

$$T_{false}(i) = s_i + (s_i - 1) + (s_i - 2) + (s_i - 2) + 14 + T(i+1).$$

Therefore,

$$T(i) \leq 4s_i + 10 + T(i+1),$$

where $s_i = 3n - 3i + 6$. Thus,

$$T(i) \leq 12(n-i) + 34 + T(i+1).$$

Writing $R(i) = T(n-i)$ then gives

$$R(i) \leq 12i + 34 + R(i-1),$$

so that $R(n) \leq \sum_{i=1}^{n}(12i + 34) = 6n^2 + 40n$. Hence, it takes at most $6n^2 + 40n$ moves to pebble the given construction for any instance of TQBF which only includes existential quantifiers.

Theorem 11. `minST` *is* `NP-Complete`*.*

Proof. We provide a reduction from `3-SAT` to `minST`. Given an instance $\mathcal{I}$ of `3-SAT` with at most n clauses or variables, we create the corresponding graph from [GLT80]. Additionally, we append a chain of length $300n^3+6n^2+40n+100$ to the graph with an edge from the sink node q_n from [GLT80] to the first node in our chain. By adding a chain of length $300n^3 + 6n^2 + 40n + 100$ we can ensure that for any *legal* pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\|}(G)$ of our graph G we have $t \geq 300n^3 + 6n^2 + 40n + 100$. By Lemma 4, at most $6n^2 + 40n$ moves are necessary to pebble the $3 - SAT$ portion of the graph, while the chain requires exactly $300n^3 + 6n^2 + 40n + 100$ moves. Thus, if $\mathcal{I}$ is satisfiable then we can find a legal pebbling $P = (P_0, \ldots, P_t)$ with space $\max_{i \leq t} |P_i| \leq 3n + 3$ and $t \leq 300n^3 + 12n^2 + 80n + 100$ moves. First, pebble the sink q_n in $t' = 6n^2 + 40n$ steps and $\max_{i \leq t'} |P_i| \leq 3n + 3$ space by Lemma 4 and then walk single pebble down the chain in $300n^3 + 6n^2 + 40n + 10$ steps keeping at most one pebble on the DAG in each step. The space time cost is at most $\mathsf{st}(P) \leq 900n^4 + 936n^3 + 276n^2+540n+300$. If $\mathcal{I}$ is not satisfiable then, by Lemma 3 for *any legal* pebbling $P = (P_0, \ldots, P_t) \in \mathcal{P}^{\|}(G)$ of our graph we have $\max_{i \leq t} |P_i| \geq 3n + 4$ and $t \geq 300n^3+6n^2+40n+100$. Thus, $\mathsf{st}(P) \geq 900n^4+1218n^3+144n^2+460n+400$ we have

$$900n^4+1218n^3+144n^2+460n+400-\big(900n^4+936n^3+276n^2+540n+300\big) > 0\,.$$

for all $n > 0$. Thus, $\mathcal{I}$ is satisfiable if and only if there exists a legal pebbling with $\mathsf{st}(P) \leq 900n^4 + 936n^3 + 276n^2 + 540n + 300$. Clearly, this reduction can be done in polynomial time, and so there is indeed a polynomial time reduction from `3-SAT` to `minST`.

We note that the same reduction from TQBF to minST fails, since there exist instances of TQBF where the pebbling time is $2^{\Omega(n)}$. However, the same relationships do hold for sequential pebbling.

The proof of Theorem 12 is the same as the proof of Theorem 11 because we can exploit the fact that the pebbling of G_{TQBF} from Lemma 4 is sequential.

Theorem 12. minSST *is* NP-Complete.

E Discussion of Other Techniques

In this section, we address a number of seemingly similar problems. In particular, several related graph partitioning problems actually have fundamentally different structures.

In the d-Vertex Separator problem, the goal is to remove the minimum number of vertices so that each remaining connected component has at most d vertices. This problem fails to translate to a successful solution for (e, d)-reducibility, as a binary tree of height $d-1$ with all directed edges pointing from parents to leaves has an exponential number of vertices, but requires no removal of vertices to ensure depth less than d.

In the d-Distance Minimum Vertex Cover problem, the goal is to find the smallest subset S of vertices so that all vertices are at most distance d from a vertex in S. Of course, even if all vertices are within distance 1 from a vertex in S, the depth of $G - S$ can be as large as $\frac{n}{2}$. Consider, for example, a path of length n, with additional edges $(i, i+2)$ for each $i \leq n-2$. Then even removing all even or all odd labeled vertices leaves a path of length $\frac{n}{2}$.

Recently, Lee [Lee17] proposed an $O(\log\ d)$-approximation algorithm to the d-Path Transversal problem (also called d-path cover by Bresar et al. [BKKS11]) for undirected graphs with parameterized complexity $2^{O(d^3 \log d)} n^{O(1)}$. The goal is to remove the minimum number of vertices so that the resulting graph contains no (undirected) paths of length d. It is not clear whether or not the techniques could be extended to deal with directed graphs. Furthermore, in most cryptanalysis applications we will have $d = \Omega(\sqrt[3]{n})$ so the approximation algorithm would run in exponential time.

Another possible approach to build an approximation algorithm for $\texttt{minREDUCIBLE}_d$ would be to exploit tree embeddings by transforming the DAG G into a tree using the longest path metric and then find a depth-reducing set for the resulting tree. However, the longest path between two vertices in directed graphs does not qualify as a metric, and it is not immediately obvious how to address this issue. Furthermore, even if one could find a tree embedding for DAGs which approximately preserves distances under the longest path metric it is not clear how to use a depth-reducing set for the tree to produce a depth-reducing set in the original DAG. In particular, observe that a complete DAG and a path of length n might both yield a simple path after performing the tree embedding under the longest path metric. However, the size of the depth-reducing sets of the two instances differ drastically.

Why Doesn't the Gilbert *et al.* Reduction Work for Cumulative Cost?

The construction from [GLT80] is designed to minimize the number of pebbles simultaneously on the graph, at the expense of larger number of necessary time steps and/or a larger average number of pebbles on the graph during each pebbling round. As a result, one can bypass several time steps by simply adding additional pebbles during some time step. For example, it may be beneficial to temporarily keep pebbles on all three nodes x_i, $\overline{x}_i$ and $\overline{x}'_i$ at times so that we can avoid repebbling the $s_i - 2$-pyramid later. Also if we do not need the pebble on node x_i for the next $\binom{s_i}{2}$ steps then it is better to discard any pebbles on x'_i and x_i entirely to reduce cumulative cost. Because we would need to repebble the s_i-pyramid later our maximum space usage will increase, but our cumulative cost would decrease. Our reduction to space-time cost works because st cost is highly sensitive to an increase in the number of pebbles on the graph even if this increase is temporary. Cumulative, unlike space cost or space-time cost, is not very sensitive to such temporary increases in the number of pebbles on the graph.

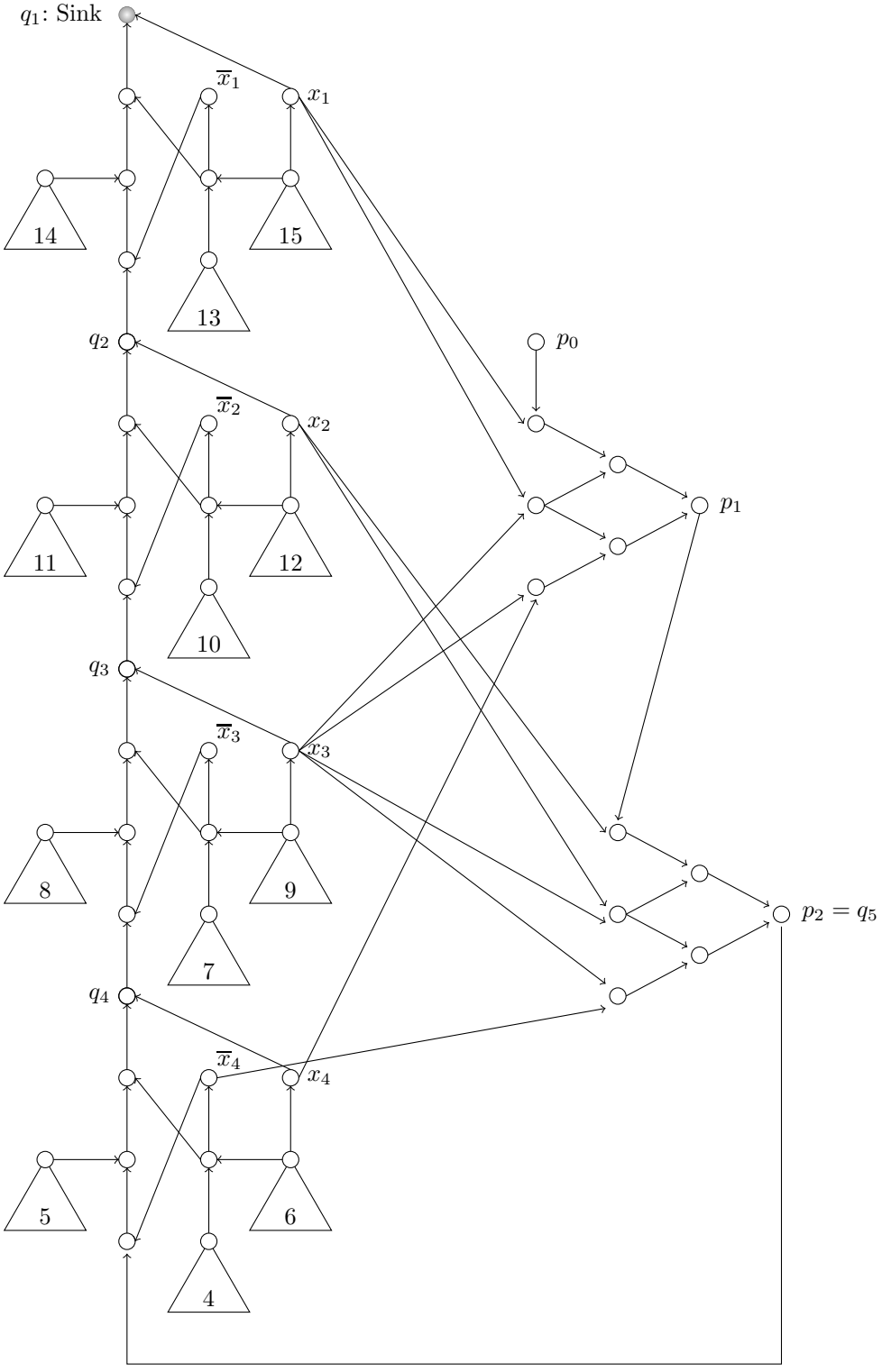

Fig. 10. Graph G_{TQBF} for $\exists x_1, x_2, x_3, x_4$ s.t. $(x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee x_3 \vee \overline{x}_4)$.