The DAG Visit Approach for Pebbling and I/O Lower Bounds

Gianfranco Bilardi

Department of Information Engineering, University of Padova, Italy

Lorenzo De Stefani[1]

Department of Computer Science, Brown University, Providence, RI, USA

Abstract

We introduce the notion of an r-visit of a Directed Acyclic Graph DAG $G = (V, E)$, a sequence of the vertices of the DAG complying with a given rule r. A rule r specifies for each vertex $v \in V$ a family of r-enabling sets of (immediate) predecessors: before visiting v, at least one of its enabling sets must have been visited. Special cases are the $r^{(top)}$-rule (or, topological rule), for which the only enabling set is the set of all predecessors and the $r^{(sin)}$-rule (or, singleton rule), for which the enabling sets are the singletons containing exactly one predecessor. The r-boundary complexity of a DAG G, $b_r(G)$, is the minimum integer b such that there is an r-visit where, at each stage, for at most b of the vertices yet to be visited an enabling set has already been visited. By a reformulation of known results, it is shown that the boundary complexity of a DAG G is a lower bound to the pebbling number of the reverse DAG, G^R. Several known pebbling lower bounds can be cast in terms of the $r^{(sin)}$-boundary complexity. The main contributions of this paper are as follows:

- An existentially tight $\mathcal{O}\left(\sqrt{d_{out} n}\right)$ upper bound to the $r^{(sin)}$-boundary complexity of any DAG of n vertices and out-degree d_{out}.
- An existentially tight $\mathcal{O}\left(\frac{d_{out}}{\log_2 d_{out}} \log_2 n\right)$ upper bound to the $r^{(top)}$-boundary complexity of any DAG. (There are DAGs for which $r^{(top)}$ provides a tight pebbling lower bound, whereas $r^{(sin)}$ does not.)
- A visit partition technique for I/O lower bounds, which generalizes the S-partition I/O technique introduced by Hong and Kung in their classic paper "I/O complexity: The Red-Blue pebble game". The visit partition approach yields tight I/O bounds for some DAGs for which the S-partition technique can only yield a trivial lower bound.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Pebbling, Directed Acyclic Graph, Pebbling number, I/O complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2022.7

Related Version *Full Version*: `https://arxiv.org/pdf/2210.01897.pdf`

Funding *Gianfranco Bilardi*: This work was supported in part by the Italian National Center for HPC, Big Data, and Quantum Computing; by MIUR, the Italian Ministry of Education, University and Research, under PRIN Project n. 20174LF3T8 AHeAD (Efficient Algorithms for HArnessing Networked Data); and by the University of Padova, under Project CPGA[3] (Parallel and Hierarchical Computing: Architectures, Algorithms, and Applications).

1 Introduction

A *visit* of a Directed Acyclic Graph (DAG) is a sequence of all its vertices. We consider different types of visits, where a type is specified by a *visit rule* r, a prescription that a vertex v can be visited only after all the vertices in one of a given family of *enabling sets* of predecessors of v have been visited. One example is the *singleton visit rule*, $r^{(sin)}$, where

[1] Corresponding author

42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2022).
Editors: Anuj Dawar and Venkatesan Guruswami; Article No. 7; pp. 7:1–7:23
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

each vertex is enabled by each singleton containing one of its predecessors. *Breadth First Search* (BFS) and *Depth First Search* (DFS) visits are special cases of $r^{(sin)}$-visits. Another example is the *topological visit rule*, $r^{(top)}$, where a vertex v is enabled only by the set of all its predecessors. The $r^{(top)}$-visits are exactly the topological orderings of the DAG. Many other rules are possible; for example, the enabling sets of a vertex could be those with a majority of its predecessors.

In this work, we investigate the *r-boundary complexity* of DAGs. The boundary complexity of G, $b_r(G)$, is the minimum integer b such that there exists an r-visit where, at each stage, for at most b of the vertices yet to be visited an enabling set has already been visited. By a reformulation of the results of Bilardi, Pietracaprina, and D'Alberto [10], in terms of the familiar concept of visit, we show that the boundary complexity of a DAG G is a lower bound to the *pebbling number* $p(G_R)$ of its reverse DAG, *i.e.,* $p(G_R) \geq b_r(G)$. The pebbling number of a DAG provides a measure of the space required by a computation with data dependences described by that DAG, in the *pebble game* framework, introduced by Friedman [18], Paterson and Hewitt [27], Hopcroft, Paul and Valiant [21]. While a pebbling game resembles a visit where each vertex can be visited multiple times, the relation between pebbling number and boundary complexity is rather subtle, as indicated by the fact that it involves graph reversal. Several pebbling lower bounds arguments in the literature (examples are mentioned in Section 3) can indeed be recast in terms of the $r^{(sin)}$-boundary complexity, thus achieving some unification in the derivation of these results. In this context, it is natural to explore the potential of the visit approach to yield significant pebbling lower bounds for arbitrary DAGs.

Main contributions. We begin our study with the singleton rule and show that, for any DAG G with n nodes and out-degree at most d_{out}, $b_{r^{(sin)}}(G) \leq 4\sqrt{d_{out}n}$. As a universal bound, this result cannot be improved, as shown by matching existential lower bounds. With respect to pebbling, there are DAGs such that $b_{r^{(sin)}}(G) = \Omega(p(G_R))$, for which the singleton rule provides asymptotically tight pebbling lower bounds. But there are also DAGs with very low $r^{(sin)}$-boundary complexity, where the reverse DAG has high pebbling number. For example, Paul, Tarjan, and Celoni [28] introduced a DAG, which we will denote as PTC, of n vertices and in-degree $d_{in} = O(1)$, and proved that $p(PTC) = \Theta(\frac{n}{\log n})$. This DAG can be easily modified to yield a DAG PTC^+ with $p(PTC^+) = \Theta(\frac{n}{\log n})$ and $b_{r^{(sin)}}((PTC^+)_R) = 2$, thus exhibiting a large gap between boundary and pebbling complexity.

It is natural to wonder whether other visit rules can lead to better bounds, whereas the singleton rule does not. We have then turned our attention to the topological rule showing that for any DAG G with n nodes and out-degree at most $d_{out} \geq 2$ the boundary $b_{r^{(top)}}(G) = \frac{d_{out}-1}{\log_2 d_{out}} \log_2 n$. This bound is existentially tight. It indicates that the potential of the topological rule for pebbling lower bounds is limited. However, the topological technique is not subsumed by the singleton one, as we exhibit DAGs for which topological visits yield a tight pebbling lower bound, whereas singleton visits yield a trivial lower bound.

For an arbitrary visit rule, r, we show that $b_r(G) \leq (d_{out} - 1)\ell + 1$, where ℓ is the length of the longest paths of G. This result is also existentially tight and is consistent with the known pebbling upper bound, $p(G_R) \leq (d_{out} - 1)\ell + 1$. It remains an open question whether a tight boundary-complexity lower bound to the pebbling number can always be found by tailoring the choice of r to the DAG, or there are DAGs for which the two metrics exhibit a gap for any rule.

We also exploit visits to analyze the *I/O complexity* of a DAG G, $IO\,(M, G)$, pioneered by Hong and Kung in [23]. This quantity is the minimum number of accesses to the second level of a two-level memory, with the first level (*i.e.,*, the cache) of size M, required to compute G.

Such computation can be modeled by a game with pebbles of two colors. Let $k(G, M)$ be the smallest integer k such that the vertices of G can be topologically partitioned into a sequence of k subsets, each with a *dominator set* and *minimum set* no larger than M. (D is a dominator of U if the vertices of U can be computed from those of D. The minimum set of U contains those vertices of U with no successor in U.) Then, $IO(G, M) \geq M(k(G, 2M) - 1)$ [23]. Dominators play a role in the red-blue game (where pebbles are initially placed on input vertices which, if unpebbled, cannot be replebbled), but not in standard pebbling (where a pebble can be placed on an input vertex at any time, hence a dominator of what is yet to be computed needs not be currently in memory). Intuitively, each segment of a computation must read a dominator set D of the vertices being computed and at least $|D| - M$ of these reads must be to the second level of the memory. It is also shown in [23] that the minimum set, say Y, of a segment of the computation must be present in memory at the end of such segment, so that at least $|Y| - M$ of its elements must have been written to the second level of the memory. In the visit perspective, the minimum set emerges as the boundary of topological visits, capturing a space requirement at various points of the computation. In addition to providing some intuition on minimum sets, this insight suggests a generalization of the partitioning technique to any type of visit. In fact, the universal upper bounds on visit boundaries mentioned above do indicate that the singleton rule has the potential to yield better lower bounds than the topological one. Following this insight, we have developed the visit partition technique. For some DAGs for which S partitions can only lead to a trivial, $\Omega(1)$, lower bound, visit partitions yield a much higher and tight lower bound.

Further related work. Since the work of Hong and Kung [23], I/O complexity has attracted considerable attention, thanks also to the increasing impact of the memory hierarchy on the performance of all computing systems, from general purpose processors, to accelerators such as GPUs, FPGAs, and Tensor engines. Their *S-partition* technique has been the foundation to lower bounds for a number of important computational problems, such as the Fast Fourier Transform [23], the definition-based matrix multiplication [3, 22, 33], sparse matrix multiplication [26], Strassen's matrix multiplication [7] (this work also introduces the "*G-flow*" technique, based on the Grigoriev flow of functions [19], to lower bound the size of dominator sets), and various integer multiplication algorithms [8, 16]. Ballard et al. [5, 4] generalized the results on matrix multiplication of [23], by means of the approach proposed by Irony, Toledo, and Tiskin in [22] based on the Loomis-Whitney geometric theorem [24], which captures a trade-off between dominator size and minimum set size. The same papers present tight I/O complexity bounds for various linear algebra algorithms for LU/Cholesky/LDLT/QR factorization and eigenvalues and singular values computation.

Four decades after its introduction, the *S-partition* technique [23] is still the state of the art for I/O lower bounds that do hold when recomputation (the repeated evaluation of the same DAG vertex) is allowed. Savage [30] has proposed the *S-span* technique, as " a slightly weaker but simpler version of the Hong-Kung lower bound on I/O time"[31]. The *S-covering* technique [10], which merges and extends aspects from both [23] and [30], is in principle more general than the S-partition technique and leads to interesting resources-augmentation considerations; however, we are not aware of its application to specific DAGs.

A number of I/O lower bound techniques have been proposed and applied to specific DAG algorithms for executions *without* recomputations. These include the *edge expansion* technique of [6], the *path routing* technique of [32], and the *closed dichotomy width* technique of [11]. While the emphasis in this paper is on models with recomputation, Section 5.4 does show how the visit partition technique specializes when recomputation is not allowed.

Automatic techniques to derive I/O lower bounds - with and without recomputation - have been developed, in part with the goal of automatic performance evaluation and code restructuring for improving temporal locality in programs, by Elango et al. [17], Carpenter et al. [13], Olivry et al. [25].

Paper organization. The visit framework is formulated in Section 2. The relationship between boundary complexity and pebbling number is discussed in Section 3. Section 4 presents universal upper bounds to the boundary complexity. Section 5 develops the visit partition technique for I/O lower bounds. Conclusions are offered in Section 6. Proofs and technical material not included in the main body due to space limitations are presented in the Appendix or in the full version of this work [9].

2 Visits of a DAG

A *Directed Acyclic Graph* (DAG) $G = (V, E)$ consists of a finite set of *vertices* V and of a set of directed *edges* $E \subseteq V \times V$, which form no directed cycle. We say that edge $(u, v) \in E$ is directed from u to v. We let $\text{pre}\,(v) = \{u \mid (u, v) \in E\}$ denote the set of predecessors of v and $\text{suc}\,(v) = \{u \mid (v, e) \in E\}$ denote the set of its successors. The maximum in-degree (resp. out-degree) of G is defined as $d_{in} = \max_{v \in V} |\text{pre}\,(v)\,|$ (resp., $d_{out} = \max_{v \in V} |\text{suc}\,(v)\,|$). Further, we denote as $\text{des}\,(v)$ (resp., $\text{anc}\,(v)$) of v's descendants (resp., ancestors), that is, the vertices that can be reached from (resp., can reach) v with a directed path. Given $V' \subseteq V$, we say that $G' = (V', E \cap (V' \times V'))$ is the sub-DAG of G *induced by* V'.

Let $\phi = (v_1, \ldots, v_i, \ldots, v_j, \ldots, v_k)$ be a sequence of vertices with $|\phi| = k$. Let $\phi[i] = v_i$, for $1 \leq i \leq j \leq k$, we denote as $\phi(i..j] = (v_{i+1}, \ldots, v_j)$ the infix from the i-th element excluded to the j-th included. If $j < i$, $\phi(i..j]$ is the empty sequence. Depending on the context, we sometimes interpret a sequence as the set of items appearing in the sequence.

A *visit* of a DAG $G = (V, E)$ is a sequence of all its vertices, without repetitions, complying with a *visit rule*:

▶ **Definition 1** (Visit rule). *A* visit rule *for a DAG $G = (V, E)$ is a function $r : V \to 2^{2^V}$ where $r\,(v) \subseteq 2^{\text{pre}(v)}$ is a non-empty family of sets of predecessors of v called* enablers *of v. The set of visit rules of G is denoted as $\mathcal{R}\,(G)$.*

Intuitively, a rule $r \in \mathcal{R}\,(G)$ permits a vertex v to be visited only after at least one of its enablers $Q \in r\,(v)$ has been entirely visited.

▶ **Definition 2** (r-sequence and r-visit). *Given a DAG $G = (V, E)$ and a visit rule $r \in \mathcal{R}\,(G)$, a sequence ψ of* distinct *vertices is an* r-sequence *of G if, for every $1 \leq i \leq |\psi|$, the prefix $\psi[1..i-1]$ includes an enabler $Q \in r\,(\psi[i])$. The r-sequences with $|\psi| = n$ are called* r-visits *and their set is denoted as $\Psi_r\,(G)$.*

Clearly, any prefix of an r-sequence is an r-sequence. Of particular interest are the " *topological visit rule*" defined as $r^{(top)}\,(v) = \{\text{pre}\,(v)\}$ and the "*singleton visit rule*" defined as $r^{(sin)}\,(v) = \{\{u\} \mid u \in \text{pre}\,(v)\}$ if $|\text{pre}\,(v)\,| > 0$ and $r^{(sin)}\,(v) = \{\emptyset\}$ otherwise.

A vertex v not contained in ψ, but enabled by some non-empty set Q included in ψ, is considered to be a "boundary" vertex.

▶ **Definition 3** (Boundary of an r-sequence). *Given a DAG $G = (V, E)$, $r \in \mathcal{R}\,(G)$, and an r-sequence ψ of G, the* r-boundary *of ψ is defined as the set:*

$$B_r\,(\psi) = \{v \in V \setminus \psi \mid \exists Q \neq \emptyset \text{ s.t. } Q \in r\,(v) \wedge Q \subseteq \psi\}\,.$$

Input vertices are never contained in the boundary of any sequence since their only enabler is the empty set.

▶ **Definition 4** (Boundary complexity). *The r-boundary complexity of an r-sequence ψ is defined as:*

$$b_r(\psi) = \max_{i \in \{1,\ldots,|\psi|\}} |B_r(\psi[1..i])|.$$

The r-boundary complexity *of G is defined as the minimum r-boundary complexity among all r-visits of G:*

$$b_r(G) = \min_{\psi \in \Psi_r(G)} b_r(\psi).$$

By definition, for any $r \in \mathcal{R}(G)$, any $\psi \in \Psi_{r^{(top)}}(G)$ and, for any $i = 1, 2, \ldots, n$ we have $B_{r^{(top)}}(\psi[1..i]) \subseteq B_r(\psi[1..i])$; thus, $b_{r^{(top)}}(\psi) \leq b_r(\psi)$. Similarly, if $\emptyset \in r(v)$ only when v has no predecessors, then $b_r(\psi) \leq b_{r^{(sin)}}(\psi)$, for any $\psi \in \Psi_r(G)$.

3 Boundary complexity and pebbling number

In this section, we discuss an interesting relationship between the pebbling number of a DAG $G = (V, E)$ and the boundary complexity of its *reverse* DAG $G_R = (V, E_R)$, where $E_R = \{(u, v) | (v, u) \in E\}$, which can prove useful in deriving pebbling lower bounds.

▶ **Theorem 5** (Pebbling lower bound). *Let G_R be the reverse of $G = (V, E)$. Then, for any $r \in \mathcal{R}(G_R)$, the pebbling number of G satisfies:*

$$p(G) \geq b_r(G_R) = \min_{\psi \in \Psi_r(G_R)} b_r(\psi).$$

In general, the analysis of the boundary complexity is simpler than the analysis of the pebbling number, in part because, in a visit, a vertex can occur only once, whereas, in a pebbling schedule, a vertex can occur any number of times.

The proof of the preceding theorem, given in Appendix, is a reformulation of a result obtained by Bilardi et al. in[10]. They introduce the *Marking Rule technique*, which is applied to DAG G rather than to its reverse. The advantage of visits over markings lies in a more direct leverage of intuition, given the widespread utilization of various kinds of visits (*e.g.*, breadth-first search, depth-first search, topological ordering) in the theory and applications of graphs.

We will explore the potential of the visit approach to yield interesting lower bounds for specific DAGs in the next section. Here, we investigate whether Theorem 5 could be strengthened by restricting the set of r-visits ψ among which $b_r(\psi)$ is minimized. The answer turns out to be negative. To clarify in what sense, we need to consider that the proof of the theorem is based on mapping each pebbling schedule π of G to an r-visit $f_r(\pi)$ of G_R, such that the boundary of each prefix of $f_r(\pi)$ is completely covered with pebbles at some stage of π. In terms of such mapping, we have:

▶ **Lemma 6** (Visit from pebbling schedule). *For any $r \in \mathcal{R}(G_R)$ and any $\psi \in \Psi_r(G_R)$, there exists a pebbling schedule π of G such that $f_r(\pi) = \psi$.*

Theorem 5 provides a general approach for obtaining pebbling lower bounds, which encompasses a number of arguments developed in the literature to analyze DAGs such as directed trees [27], pyramids [29] and stacks of superconcentrator [21]. A reformulation of these arguments within the visit framework can be found in [15] stacks of superconcentrators and in [9, Section 3.1] for q-pyramid and q-complete trees DAGs.

4 Upper bounds on boundary complexity

It is natural to wonder whether the pebbling lower bound of Theorem 5 is tight. As we will see in this section, both the singleton and the topological rules, while providing tight bounds for some DAGs, yield weak lower bounds for others. Whether a tight lower bound could be obtained for any DAG G, by tailoring the visit rule to G, does remain an open question.

In particular, we will establish universal upper bounds on the boundary complexity of any DAG, with respect to any rule, in terms of outdegree and depth. We will also establish (different) universal upper bounds for both the singleton and the topological rule in terms of outdegree and the number of vertices. Before presenting these results, we introduce the notion of enabled reach, a particular set of vertices associated with a vertex v, in the context of a partial visit that includes v. This concept will play a role in the derivation of each of the three universal upper bounds.

4.1 The *enabled reach* of a vertex

In the construction of a visit sequence, we will use a divide and conquer approach whereby, having constructed a prefix ψ of the sequence, the next segment, ϕ, of the sequence is obtained by visiting a suitably chosen sub-DAG, $G' = (V', E')$, according to an appropriate rule r'. It is useful for the boundary of G' to be "self-contained" in the sense that its visit does not generate any boundary outside V'. If this is the case, the boundary of $\psi\phi$ will be a subset of the boundary of ψ, so that the visit of G' contributes to the reduction of both the set of vertices yet to be visited and the current boundary. The enabled reach, a set of vertices introduced next, induces a sub-DAG G' with the desired properties.

▶ **Definition 7** (Enabled reach). *Let $G = (V, E)$ and $r \in \mathcal{R}(G)$. Given an r-sequence ψ and a vertex $v \in \psi$, the r-enabled reach of v given ψ is the set:*

$$\mathrm{reach}_r(v|\psi) = \{u \mid \exists\ \psi' u \subseteq \mathrm{des}(v) \ \textit{s.t. } \psi\psi' u \textit{ is an } r\textit{-sequence}\}.$$

Intuitively, we can think of the r-enabled reach of a vertex v given an r-sequence ψ as the set of all the descendants of v which can be visited by extending ψ only with descendants of v. As an example, for $r^{(sin)}$, we have that the $r^{(sin)}$-enabled reach of a vertex v given a ψ corresponds to the set of the descendants of v not in ψ. The enabled reach exhibits the following crucial property:

▶ **Lemma 8** (Enabled Reach Property). *Given $G = (V, E)$, $v \in V$ and $r \in \mathcal{R}(G)$, let ψ be an r-sequence including v. Let $G' = (V', E')$ be the sub-DAG induced by $V' = \mathrm{reach}_r(v|\psi)$. Let $r' \in \mathcal{R}(G')$ be such that $r'(v) = \{Q \setminus \psi | Q \in r(v) \wedge Q \setminus \psi \subseteq V'\}$, for all $v \in V'$. If $\psi' \in \Psi_{r'}(G')$ then (a) $\psi\psi'$ is a r-sequence of G; (b) for any $i = 1, \ldots |\psi'|$, $B_r(\psi\psi'[1..i]) \subseteq B_r(\psi) \cup B_{r'}(\psi'[1..i])$; and (c) $b_r(\psi\psi') \leq b_r(\psi) + b_{r'}(\psi')$.*

Proof. By definition, $Q' \in r(v)$ if an only is there exists $Q \in r(v)$ such that $Q' = Q \setminus \psi$. By construction, for any $1 \leq j \leq |\psi'|$, a vertex v appears in $\psi'[1..j]$ if there exists $Q' \in r'(v)$ such that $Q' \in \psi'[1..j]$ which implies there exists $Q \in \psi\psi'[1..j]$, and, thus, $\psi\psi'[1..j]$ is an r-sequence.

Recall that a vertex appears in the boundary of a r sequence if it is enabled but not visited. By construction, $\psi' = \mathrm{reach}(v|\psi)$, thus, by definition, any vertex which is enabled by a subset of $\psi\psi''$ must either be included in $B_r(\psi)$ or must be included among the vertices of $\mathrm{reach}(v|\psi)$ not yet visited in $\psi\psi'[1..j]$ and enabled by ψ', that is $B_{r'}(\psi'[1..i])$. Hence, we have that $b_r(\psi\psi') \leq b_r(\psi) + b_{r'}(\psi')$. ◀

Lemma 8 states that visiting the r-enabled reach of a vertex v given a r-sequence ψ of G does not enable any vertex outside $\text{reach}_r(v|\psi)$ which was not enabled by ψ alone. Therefore, once the sub-DAG induced by $\text{reach}_r(v|\psi)$ is visited, the only vertices left in the r-boundary are those enabled by ψ that have not been visited thus far.

The enabled reach will be a key ingredient in the construction of visits in the next three subsections. The choice of both r-sequence ψ and of vertex v has to be tailored to the particular r. Also, highly influenced by r are the size of the boundary of ψ and the reduction achieved by G' in the parameters (*e.g.*, depth or number of vertices) governing the boundary complexity, hence the shape of the resulting bound.

4.2 General rules

The *topological depth* of a DAG is the length (*i.e.,* number of edges) of its longest directed paths. The boundary complexity, according to any visit rule, can be bounded in terms of the depth and the out-degree. The basic property that is exploited is that if b is a successor of a, then the depth of the sub-DAG induced by the descendants of b is smaller than the depth of the sub-DAG induced by the descendants of a.

▶ **Theorem 9** (Topological-depth general boundary complexity upper bound)**.** *Consider $G = (V, E)$ with maximum out-degree d_{out} and topological depth ℓ. For any visit rule $r \in \mathcal{R}(G)$, there exists an r-visit $\psi \in \Psi_r(G)$ such that $b_r(\psi) \leq (d_{out} - 1)\,\ell + 1$.*

Proof of Theorem 9 is presented in the appendix. This upper bound is *existentially tight*: For some visit rules r, there exist some DAGs for which $b_r(G) = \Theta(d_{out}\ell)$. An example is given by the reverse q-pyramid DAG, discussed in the extended version of this work [9, Section 3.1], for which $b_{r^{(sin)}}(G) = \Theta(d_{out}\ell) = \Theta\left(\sqrt{qn}\right)$, since $d_{out} = q$ and $\ell = \sqrt{n/q}$.

Below, we derive universal upper bounds for the singleton and the topological rule, which are expressed in terms of n and d_{out}. These bounds are tighter than that of Theorem 9 for DAGs with ℓ suitably large (as a function of n and d_{out}).

4.3 Singleton rule $r^{(sin)}$

The $r^{(sin)}$ rule has the interesting property that the enabled reach of v, given ψ, contains all the descendants of v not in ψ. One can easily find a v with suitably few descendants, say, less than $n/2$. A $r^{(sin)}$-sequence ψ that contains v can be obtained as the sequence of vertices on a path from an input to v. If this path has length k, then its boundary could be of a size as big as $(d_{out} - 1)k + 1$; therefore, a small k is a prerequisite to guaranteeing small boundary complexity. In general, a good enough upper bound to k cannot be guaranteed for the entire DAG. However, it is possible to partition the DAG into a sequence of "blocks" such that (i) blocks can be visited one at a time in the order they appear in the sequence; (ii) there is a reasonably small upper bound ($O(\sqrt{d_{out}n})$) on the number of nodes that are enabled by the nodes in a block, but lie outside the block, and they lie all in the next block; and (iii) in each block, each node is reachable from one of the block inputs by a path of reasonably small length ($O(\sqrt{d_{out}n})$). When the details are filled in, the outlined approach yields the following results.

▶ **Theorem 10.** *Given an $G = (V, E)$ with $|V| = n$ and maximum out-degree at most d_{out} there exists a visit $\psi \in \Psi_{r^{(sin)}}(G)$ s.t. $b_{r^{(sin)}}(\psi) \leq 4\left(\sqrt{2}+1\right)\sqrt{d_{out}n}$.*

Proof. For $d_{out} = 0$, all vertices in G are isolated and, having no predecessors, are enabled only by the empty set. Thus, no vertex belongs to the $r^{(sin)}$-boundary of any $r^{(sin)}$-visit ψ, hence $b_{r^{(sin)}}(\psi) = 0$, and the stated bound holds. In the sequel, we assume $d_{out} \geq 1$, and proceed by induction on n.
Base: For $n = 1$, that is, $G = (\{u\}, \emptyset)$, the statement is trivially verified as v is an input vertex and the only $r^{(sin)}$-visit, $\psi = u$, has $r^{(sin)}$-boundary complexity zero.
Inductive step $(n \geq 2)$*:*

Case 1: $|I| > 1$**.** Here, no vertex v is an ancestor of all vertices. Let $v \in I$ and let ψ' be a $r^{(sin)}$-visit of the DAG induced by $\mathrm{des}\,(v)$, with boundary complexity at most at most $c\sqrt{d_{out}n}$, which does exist by the inductive hypothesis, since $|\mathrm{des}\,(v)\,| < n$. Similarly, let ψ'' be a $r^{(sin)}$-visit of the DAG induced by $V \setminus \mathrm{des}\,(v)$, with boundary complexity at most $c\sqrt{d_{out}n}$. Clearly, $\psi = \psi'\psi'' \in \Psi_{r^{(sin)}}\,(G)$. By the definition of enabled reach, for $r^{(sin)}$, we have that $\mathrm{reach}_{r^{(sin)}}\,(v|v) = \mathrm{des}\,(v) \setminus \{v\}$. Hence, by Lemma 8, $B_{r^{(sin)}}(\psi') = \emptyset$. The stated bound follows.

Case 2: $|I| = 1$**.** We partition V into non-empty "*levels*" $L\,(1)\,, \ldots, L\,(\ell_s)$, such that $v \in L(i)$ if and only if the shortest directed path from u to v has length i. This path is also a $r^{(sin)}$-sequence. Further, the r-boundary of any $r^{(sin)}$-sequence of G included in the first i levels is a subset of the first i+1 levels.

We say that level $L(i)$ is a *bottleneck* if $|L(i)| \leq \gamma\sqrt{d_{out}n}$ and let $i_1 < i_2 < \ldots < i_k$ denote the indices of the bottlenecks. Here, $\gamma > 0$ is a constant, whose value will be determined in the course of the proof. Conventionally, we also let $i_{k+1} = \ell_s$+1 and $L(\ell_s + 1) = \emptyset$. Since $L(1) = \{u\}$, we have that $i_1 = 1$. We group consecutive levels into *blocks* $V_j = \cup_{i_j \leq l < i_{j+1}} L\,(l)$, for $j = 1, 2, \ldots, k$, so that the j-th block begins with the j-th bottleneck and ends just before the (j+1)-st one, or with the last level, $L(\ell_s)$, if $j = k$. The number of levels of a block is upper bounded as $i_{j+1} - i_j \leq \frac{\sqrt{n}}{\gamma\sqrt{d_{out}}}$.

We construct an $r^{(sin)}$-visit of the form $\psi = \psi_1\psi_2 \ldots \psi_k$, where ψ_j is a $r^{(sin)}$-visit of the sub-DAG G_j induced by block V_j. Since the V_j's partition V, $\psi \in \Psi_{r^{(sin)}}\,(G)$. By the properties of the levels mentioned above, $B_{r^{(sin)}}(\psi_1 \ldots \psi_{j-1}) = L\,(i_j)$. Furthermore, as $V_1, \ldots, V_{j-1}$ have already been visited by $\psi_1 \ldots \psi_{i-1}$ and ψ_j is a $r^{(sin)}$-visit of G_j, any of its prefixes ψ'_j may only enable vertices in V_j (*i.e.*, the boundary of ψ'_j in G_j) and vertices in $L(i_{j+1})$ (*i.e.*, the children of vertices in $\psi'_j \subseteq V_j$, which are not in $V_1 \cup \ldots \cup V_j$). Therefore,

$$
\begin{aligned}
b_r^{(sin)}(\psi) &\leq \max_j\{|L\,(i_j)\,| + |L\,(i_{j+1})\,| + b_{r^{(sin)}}\,(\psi_j)\} \\
&\leq 2\gamma\sqrt{d_{out}n} + \max_j\{b_{r^{(sin)}}\,(\psi_j)\}.
\end{aligned}
$$

A case analysis shows how each ψ_j can be chosen so that the above term is at most $c\sqrt{d_{out}n}$.

Case 2.1. $|V_j| \leq n/2$. By the inductive hypothesis, there exists $\psi_j \in \Psi_{r^{(sin)}}\,(G_j)$ such that $b_{r^{(sin)}}(\psi_j) \leq c\sqrt{d_{out}n/2} = \frac{c}{\sqrt{2}}\sqrt{d_{out}n}$. A sufficient condition for the desired result is that $2\gamma + \frac{c}{\sqrt{2}} \leq c$, which we will discuss below.

Case 2.2. $|V_j| > n/2$. Let $u_1, u_2, \ldots, u_d$ be the input vertices of G_j.

Case 2.2.a. No input of G_j has more than $n/2$ descendants, in G_j. Then, we construct an $r^{(sin)}$-visit of G_j as $\psi_j = u_1\psi_{u_1}u_2\psi_{u_2} \ldots, u_d\psi_{u_d}$, where ψ_{u_l} is a $r^{(sin)}$-visit, with minimum boundary complexity, of the sub-DAG induced by the descendants of u_l in G_j which have not been visited in $u_1\psi_{u_1}u_2\psi_{u_2}\ldots,u_l$. This is the $r^{(sin)}$-enabled reach of u_l in G_j, given $u_1\psi_{u_1}u_2\psi_{u_2}\ldots,u_l$. Since the input vertices of G_j are not in the boundary of any prefix of ψ_j, by Lemma 8,

$$b_r^{(sin)}(\psi_j) \leq \max_{l=1,\ldots,d} b_{r^{(sin)}}(\psi_{u_j}) \leq c\sqrt{d_{out}\frac{n}{2}},$$

where the last step follows by the inductive hypothesis, considering that $\psi_{u_j} \subseteq \mathrm{des}\,(u_j)$ and, by the assumption of this case, $|\mathrm{des}\,(u_j)| \leq n/2$ (here, and throughout the rest of this proof, $\mathrm{des}\,(v)$ refers to the descendants of v in G_j). The sufficient condition for the desired result is the same as in Case 2.1

Case 2.2.b. There is an input of G_j, w.l.o.g, say u_1, such that $|\mathrm{des}\,(u_1)| > n/2$. In order to break down V_j into pieces of size smaller than $n/2$, to be visited one at a time, we select a vertex $y \in \mathrm{des}\,(u_1)$ such that $|\mathrm{des}\,(y)| \geq n/2$ and $\max_{z\in \mathrm{suc}(y)} |\mathrm{des}\,(z)| < n/2$. Specifically, we can choose y as the last vertex, in a topological ordering of G_j, with at least $n/2$ descendants. Let $y \in L(i)$, where clearly $i_j \leq i < i_{j+1}$, and consider a shortest path πy among those from input vertices of G_i to y. We have $|\pi| = i - i_j < i_{j+1} - i_j \leq \frac{\sqrt{n}}{\gamma\sqrt{d_{out}}}$.

Consider now the visit $\psi_j = \pi y\psi_y\psi'$ such that ψ_y is a $r^{(sin)}$-visit of the sub-DAG induced by $V_j \cap \mathrm{reach}_{r^{(sin)}}\,(y|\pi y)$ constructed as discussed in Case a, and ψ' is a $r^{(sin)}$-visit, with minimum boundary complexity, of the sub-DAG induced by $V_j \setminus \pi y\psi_y$. The boundary associated with any prefix of π includes at most d_{out} successors for each of its vertices, which are at most $|\pi| \leq \frac{\sqrt{n}}{\gamma\sqrt{d_{out}}}$. Thus, the total contribution of π to the boundary is at most $\frac{\sqrt{d_{out}n}}{\gamma}$. Arguing along the lines of Case a, it can be shown that $b_{r^{(sin)}}(\psi_y) \leq \frac{c}{\sqrt{2}}\sqrt{d_{out}n}$. Finally, since $|\psi_{v^*}| \geq n/2$, then $|V_j \setminus \pi y\psi_y| < n/2$. Hence, by the inductive hypothesis, $b_{r^{(sin)}}\,(\psi') < \frac{c}{\sqrt{2}}\sqrt{d_{out}n}$.

By Lemma 8, we can conclude that $\psi_j \in \Psi_{r^{(sin)}}\,(G_j)$ and

$$b_{r^{(sin)}}(\psi_j) \leq \frac{1}{\gamma}\sqrt{d_{out}n} + \max\{b_{r^{(sin)}}\,(\psi_y)\,, b_{r^{(sin)}}\,(\psi')\} \leq \left(\frac{1}{\gamma} + \frac{c}{\sqrt{2}}\right)\sqrt{d_{out}n}.$$

To establish the stated result, we need to satisfy the bound $2\gamma + \frac{1}{\gamma} + \frac{c}{\sqrt{2}} \leq c$. This requirement is more stringent than the one for Case 2.1 and Case 2.2.a. Solving for c, the sufficient condition is $c \geq \sqrt{2}\left(\sqrt{2}+1\right)\left(2\gamma + \frac{1}{\gamma}\right)$. The r.h.s. is minimized when we let $\gamma = \frac{1}{\sqrt{2}}$, which yields $c \geq 4\left(\sqrt{2}+1\right)$. ◀

The upper bound in Theorem 10 is existentially tight as there exist DAGs, such as q-pyramids discussed in [9, Section 3.1], of matching $r^{(sin)}$-boundary complexity.

4.4 Topological rule $r^{(top)}$

The following property is peculiar to the $r^{(top)}$-enabled reach:

▶ **Lemma 11** (Disjointness of enabled $r^{(top)}$-reach). *Let ψ be a $r^{(top)}$-sequence of DAG $G = (V, E)$ and let $u, v \in b_{r^{(top)}}\,(\psi)$ be distinct vertices. Then,* $\mathrm{reach}_{r^{(top)}}\,(u|\psi u) \cap \mathrm{reach}_{r^{(top)}}\,(v|\psi v) = \emptyset$.

The proof is presented in the Appendix. The disjointedness of the enabled-reach sets ensures that, if there are k vertices in the boundary, at least one of them has an enabled reach with fewer than n/k vertices. By leveraging this property, we obtain the following result.

▶ **Theorem 12** (Upper bound boundary complexity $r^{(top)}$-visits). *For any DAG $G = (V, E)$, there exists a visit $\psi \in \Psi_{r^{(top)}}(G)$ such that (a) if $d_{out} = 0$, then $b_{r^{(top)}}(\psi) = 0$; (b) if $d_{out} = 1$, then $b_{r^{(top)}}(\psi) = 1$; and (c) if $d_{out} \leq D$, for some $D \geq 2$, then $b_{r^{(top)}}(\psi) \leq \frac{D-1}{\log_2 D} \log_2 n + 1$.*

The proof of Theorem 12 is presented in the Appendix. The upper bound in Theorem 12 is existentially tight as there exist DAGs such as inverted q-trees (discussed in the extended version of this work [9, Section3.1]) with matching $r^{(top)}$-boundary complexity.

Interestingly, there exist DAGs for which the $r^{(top)}$-boundary complexity is asymptotically higher than the $r^{(sin)}$-boundary complexity. One such DAG G is shown in Figure 1. It is a simple exercise to prove that $b_r^{(sin)}(G) = 1$, $b_r^{(top)}(G) = \Theta(\log n)$, and $p(G_R) = \Theta(\log n)$.

5 Visits and I/O complexity

In this section, we show how the visit framework is fruitful in the investigation not just of space complexity but also of I/O complexity, by developing a new I/O lower bound technique, named "*visit partition*". This extends a result by Hong and Kung [23], which has provided the basis for many I/O lower bounds in the literature thus far.

The I/O model of computation is based on a system with a memory hierarchy of two levels: a fast memory or *cache* of M memory words and a *slow memory*, with an unlimited number of words. We assume that any value associated with a DAG vertex can be stored within a single memory word. A *computation* is a sequence of steps of the following types: (i) *operations*, with operands and results in cache; (ii) *reads*, that copy the content of a memory location into a cache location; and (iii) *writes*, that copy the content of a cache location into a memory location. Reads and writes are also called I/O operations. Input values are assumed to be available in the slow memory at the beginning of the computation. Output values are required to be in the slow memory at the end of the computation. The conditions under which a computation is a valid execution of a given DAG are intuitively clear. They can be formalized in terms of the "*red-blue pebble game*", introduced by Hong and Kung in [23]. The *I/O complexity* $IO(G, M)$ of a DAG G is defined as the minimum number of I/O operations over all possible computations of G. The *I/O write complexity* $IO_{\mathcal{W}}(G, M)$ and the *I/O read complexity* $IO_{\mathcal{R}}(G, M)$ are similarly defined.

The visit partition approach to I/O lower bounds develops along the following lines:

- A visit rule r is chosen for the analysis.
- A procedure is specified to map each computation ϕ of the given DAG G to an r-visit ψ of the reverse DAG G^R.
- Given any partition of visit ψ into consecutive segments, a set of I/O operations is identified for each segment, the sets of different segments being disjoint, whence their contributions can be added in the I/O lower bound.
- The number of read operations associated with a segment is lower bounded in terms of the size of a minimum post-dominator of the segment.
- The number of write operations associated with a segment is lower bounded in terms of the number of segment vertices that either inputs of G_R (hence, outputs of G), or belong to the boundary of the visit at the beginning of the segment. The resulting global lower bound, modified by the addition of the term $|I| - |O|$, also applies to read operations.
- A lower bound, q, to the I/O of a given DAG, can be established by showing that, *for each* visit, *there exists* a segment partition that requires at least q I/O operations, between reads and writes.

The technical details of this outline are presented in the next subsections.

5.1 Segment partitions of a visit

The concept of *post-dominator set* mirrors that of *dominator set* used in [23]:

▶ **Definition 13** (Post-dominator set). *Given $H = (W, F)$ and $X \subseteq W$, a set $P \subseteq W$ is a* post-dominator *set of $V' \subseteq W$ if every directed path from a vertex in V' to an output vertex intersects P. We denote as $pd_{min}(X)$ the minimum size of any post-dominator set of X.*

It is simple to see that P is a post-dominator set of X in H if and only if P is a dominator set of X in the reverse DAG H_R.

Let ψ be an r-visit of $G_R = (V, E_R)$. A *segment partition* of ψ into k *segments* is identified by a sequence of indices $\mathbf{i} = (i_1, i_2, \ldots, i_k)$, with $1 \leq i_1 < i_2 < \ldots < i_k = n$. We also let $i_0 = 0$, for convenience. Since ψ is a permutation of the vertices in V, the segments partition V. For $1 \leq j \leq k$, $\psi(i_{j-1}..i_j]$ is called the j-th *segment* of the partition. Two measures play a role in our I/O lower bound analysis of any segment:

- The size, $pd_{min}(\psi(i_{j-1}..i_j])$, of minimum post-dominator sets of $\psi(i_{j-1}..i_j]$.
- The r-*entering boundary size*

$$b_r^{(ent)}(\psi(i_{j-1}..i_j]) = |B_r^{(ent)}(\psi(i_{j-1}..i_j])|$$

where, denoting as I_R the set of input vertices of G_R,

$$B_r^{(ent)}(\psi(i_{j-1}..i_j]) = (I_R \cup B_r(\psi[1..i_{j-1}])) \cap \psi(i_{j-1}..i_j].$$

5.2 Lower bound

Let $\phi = (\phi_1, \phi_2, \ldots, \phi_T)$ be a computation of a DAG $G = (V, E)$ in the I/O model, in T steps, where ϕ_t is the t-th step. Given an $r \in \mathcal{R}(G_R)$, we construct an r-visit ψ of G_R corresponding to ϕ. Our I/O lower bounds will be based solely on properties of the visit. To construct ψ, the computation is examined backward, one step at a time. The visit is constructed incrementally, by extending an initially empty prefix. Let $\psi[1..i(t)]$ be the prefix already constructed just before processing computation step ϕ_t (initially, $i(T) = 0$). For $t = T, T{-}1, \ldots 1$, if ϕ_t is either a functional operation evaluating vertex $v \in V \setminus I$ or a read operation copying a vertex $v \in I$ (input of G) into the cache, then, if v has not already been visited (*i.e.*, $v \notin \psi[1..i(t)]$) and at least one enabler set $Q \in r(v)$ has already been visited (*i.e.*, $Q \subseteq \psi[1..i(t)]$), then v is added to the visit (*i.e.*, $i(t{-}1) = i(t){+}1$ and $\psi[1..i(t{-}1)] = \psi[1..i(t)]v$). Otherwise, the visit constructed thus far remains unchanged (*i.e.*, $i(t-1) = i(t)$).

By construction, a vertex is included in ψ at most once and only after at least one of its enablers has been visited. To conclude that ψ is indeed an r-visit of G_R it remains to show that it contains all the vertices. The vertices in I_R (which are the inputs of G_R, that is, the outputs of G) are added to the visit when they are first encountered in the backward processing of ϕ, since they are enabled by the empty set. Suppose now, by contradiction, that there are vertices in $V \setminus I_R$ that are not included in ψ. Then, let t_v be the smallest t such that v is computed (if $v \in V \setminus I$) or read from slow memory (if $v \in I$) in step ϕ_t, and let u be the vertex with the largest t_u that is not in ψ. It must be the case that if $\psi[1..i(t_u)]$ does not include any enabler of u, which in turn implies that there exists a predecessor w of u (in G_R) that does not belong to $\psi[1..i(t_u)]$. Since $t_w > t_u$, this implies that $\psi[1..i(t_w)]$ does not include any enabler of w, whence $w \notin \psi$, which contradicts the definition of u.

The procedure just described to construct an r-visit of G_R from an I/O computation ϕ of G is quite similar to the one in the proof of Theorem 5 for the analysis of the pebbling number. The differences are due to the circumstance that the standard I/O model assumes the inputs to be initially available in slow memory, whereas the pebbling model assumes that the inputs can be (repeatedly) loaded into the working space at any time.

▶ **Lemma 14** (Visit partition). *Let ϕ be a computation of DAG $G = (V, E)$ on the I/O model with a cache of M words. Let $r \in \mathcal{R}(G_R)$ and let ψ be the r-visit of G_R constructed from ϕ, as described above, and $\mathbf{i} = (i_1, i_2, \ldots, i_k)$ any of its segment partitions. Then, the number $IO_{\mathcal{W}}(\phi, M)$ of write I/O operations and the number $IO_{\mathcal{R}}(\phi, M)$ of read I/O operations executed by ϕ satisfy the bounds*

$$IO_{\mathcal{W}}(\phi, M) \quad \geq \quad \mathcal{W}_r(\mathbf{i}, \psi, M) := \sum_{j=1}^{k} \max\{0, b_r^{(ent)}\left(\psi(i_{j-1}..i_j]\right) - M\}, \tag{1}$$

$$IO_{\mathcal{R}}(\phi, M) \quad \geq \quad \mathcal{W}_r(\mathbf{i}, \psi, M) + |I| - |O|, \tag{2}$$

$$IO_{\mathcal{R}}(\phi, M) \quad \geq \quad \mathcal{R}_r(\mathbf{i}, \psi, M) := \sum_{j=1}^{k} \max\{0, pd_{min}\left(\psi(i_{j-1}..i_j]\right) - M\}. \tag{3}$$

The total number $IO(\phi, M) = IO_{\mathcal{W}}(\phi, M) + IO_{\mathcal{R}}(\phi, M)$ of I/O operations executed by ϕ satisfies the bound

$$IO(\phi, M) \geq \mathcal{W}_\psi(\mathbf{i}, M) + \max\{\mathcal{R}_\psi(\mathbf{i}, M), \mathcal{W}_\psi(\mathbf{i}, M) + |I| - |O|\}. \tag{4}$$

Proof. We will analyze the entering boundary and the post-dominator contributions to the lower bound on the number of, respectively, write and read I/O operations for a generic segment of the visit $\psi(h..i]$, with $0 \leq h < i \leq n$, and then compose the contributions of the segments in partition $\mathbf{i}$. For $l = 1, \ldots, n$, we let τ_l be such that $\psi[l]$ has been added to visit ψ in correspondence of computation step ϕ_{τ_l}. Observe that $\tau_1 > \tau_2 > \ldots > \tau_n$, since the visit is constructed from the computation in reverse. When speaking of cache or of the slow memory at time t, we refer to their state just before the execution of computation step ϕ_t.

Proof of (1) **– Boundary bound.** We claim that, at time τ_h, the value of each vertex of set $B_r^{(ent)}\left(\psi(h..i]\right)$ is stored in cache or in slow memory. Let

$$v \in B_r^{(ent)}\left(\psi(h..i]\right) = \left(B_r\left(\psi\left[1..h\right]\right) \cup O\right) \cap \psi(h..i],$$

where $O = I_R$ is the set of output vertices of G (also, input vertices of G_R). Let $v = \psi[g]$, with $h < g \leq i$.

Case 1. If $v \in O$, then at the time $\tau_g \in [\tau_i, \tau_h)$ when it has been visited v has been computed for the last time. Thereafter, by the rules of the I/O model, the value of v must be kept in memory till the end of the computation. At time τ_h, v can be either in cache or slow memory, but at some time $t > \tau_g$ it must be written in slow memory.

Case 2. If $v \in B_r\left(\psi\left[1..h\right]\right)$, then $\psi\left[1..h\right]$ includes an r-enabler of v and we consider the smallest index f such that $\psi\left[1..f\right])$ includes an r-enabler of v. Clearly $f \leq h$ and $u = \psi[f]$ is a successor of v in G. We argue that the value of v must be in memory during the interval $(\tau_g, \tau_f]$. In fact, when u is computed, v must be in cache. We separately analyze two subcases.

Case 2.1. If $v \in V \setminus I$, then it is not computed at any time $t \in (\tau_g, \tau_f]$, otherwise v would be visited at t. Since $\tau_g < \tau_h \leq \tau_f$, we have that (a copy of the value of) v is in memory at time τ_h.

Case 2.2. If $v \in I$, then it is not read from slow memory at any time $t \in (\tau_g, \tau_f]$, otherwise v would be visited at t. Then, throughout this interval, v must be kept in cache. Since $\tau_g < \tau_h \leq \tau_f$, we have that (a copy of the value of) v is in cache at time τ_h.

Given that at most M vertices of $B_r^{(ent)}(\psi(h..i])$ can be in cache at τ_h, we conclude that at least $\max\{0, |B_r^{(ent)}(\psi(h..i])| - M\} = \max\{0, b_r^{ent}(\psi(h..i]) - M\}$ vertices must be in slow memory. Such vertices must all fall under Case 2.1 since the vertices in Case 2.2 must be in the cache. Then, they must have been written into slow memory, thus contributing to the number of write I/O, like the vertices in O (Case 1). Moreover, the contributions of the (disjoint) segments of partition $\mathbf{i}$ can be added, since they count write operations involving vertices that belong to disjoint sets. This concludes the proof for (1).

Proof of (2) **– Modified Boundary bound.** By examining the argument for Case 2.1, we see that the vertices involved must also be read, at some time $t \geq \tau_h$, so that, those that are not in cache at time τ_h contribute to the number of read I/O. Each vertex in I is read at least once from slow memory. Finally, considering that no read I/O has been argued when $v \in O$, we reach (2).

Proof of (3) **– Post-dominator bound.** We claim that the set Y of vertices that are in cache at time τ_i or are read into the cache during the interval $[\tau_i, \tau_h)$ is a dominator set of $\psi(h..i]$ in G or, equivalently, a post-dominator set of $\psi(h..i]$ in G_R.

Let $v = \psi[g]$, with $h < g \leq i$. If $v \in I$, then at the time $\tau_g \in [\tau_i, \tau_h)$, when v has been visited, it has been read into the cache, so that $v \in Y$, whence v is dominated by Y.

If $v \in V \setminus I$, then at the time $\tau_g \in [\tau_i, \tau_h)$ when v has been visited it has been computed. If, by way of contradiction, v is not dominated by Y, then there is a directed path in G, say $(v_1, v_2, \ldots v_q = v)$, with no vertex in Y. Let v_s be the first vertex on this path computed during interval $[\tau_i, \tau_h)$, which must exist since v is computed during such interval. When v_s is computed, v_{s-1} must be in cache, since it is one of its operands. However, during $[\tau_i, \tau_h)$, v_{s-1} cannot be in cache since is not computed (by the definition of v_s), nor is it available at the initial time τ_i or read from slow memory (since $v_s \notin Y$). Thus, we have reached a contradiction, which shows that v is actually dominated by Y.

Given that at most M vertices of Y can be initially in the cache, we conclude that at least $\max\{0, |Y| - M\} \geq \max\{0, pd_{min}(\psi(h..i]) - M\}$ must be brought into cache by read operations that occur during the interval $[\tau_i, \tau_h)$. Moreover, the contributions of the (disjoint) segments of partition $\mathbf{i}$ can be added since they count read operations occurring in different time intervals. This concludes the proof for (3).

Finally, a straightforward combination of bounds (1), (2), and (3) yields bound (4). ◀

We observe that, in Lemma 14, we can choose the visit rule and then, for the visit ψ corresponding to a given computation ϕ, we can choose the segment partition $\mathbf{i}$ with the goal of maximizing the resulting lower bound for the cost metric of interest. However, for the lower bound to apply to (all computations of) DAG G, we have to consider the minimum lower bound over all visits. The preceding observations are made more formal in the next theorem. It is generally possible that none of the computations that minimize the number of read operations also minimizes the number of write operations, so that the I/O complexity may be larger than the sum of the read and of the write complexity.

▶ **Theorem 15** (I/O lower bound)**.** *Given a DAG $G = (V, E)$, with input set I and output set O, a visit rule $r \in \mathcal{R}(G_R)$, a visit of G_R according to this rule $\psi \in \Psi_r(G_R)$, and a cache size M, we define the quantities*

$$\mathcal{W}_r(\psi, M) = \max_{\mathbf{i}\in\mathcal{I}(\psi)} \mathcal{W}_r(\mathbf{i}, \psi, M), \tag{5}$$

$$\mathcal{R}_r(\psi, M) = \max_{\mathbf{i}\in\mathcal{I}(\psi)} \mathcal{R}_r(\mathbf{i}, \psi, M). \tag{6}$$

where $\mathcal{I}(\psi)$ denotes the set of all segment partitions of ψ and the quantities $\mathcal{W}_\psi(\mathbf{i}, M)$ and $\mathcal{R}_\psi(\mathbf{i}, M)$ are those introduced in Equations (1) *and* (3)*, respectively. Then, the write I/O complexity $IO_{\mathcal{W}}(G, M)$, the read I/O complexity $IO_{\mathcal{W}}(G, M)$, and the total I/O complexity $IO(G, M)$ satisfy the following bounds:*

$$IO_{\mathcal{W}}(G, M) \geq \min_{\psi\in\Psi_r(G_R)} \mathcal{W}_r(\psi, M), \tag{7}$$

$$IO_{\mathcal{R}}(G, M) \geq \min_{\psi\in\Psi_r(G_R)} \max\{\mathcal{R}_r(\psi, M),\ \mathcal{W}_r(\psi, M) + |I| - |O|\}, \tag{8}$$

$$IO(G, M) \geq \min_{\psi\in\Psi_r(G_R)} \{\mathcal{W}_r(\psi, M) + \max\{\mathcal{R}_r(\psi, M), \mathcal{W}_r(\psi, M) + |I| - |O|\}\}. \tag{9}$$

Proof. Bound (1) of Lemma 14 applies to a computation ϕ of G for which the procedure constructing the visit outputs ψ. The bound holds for any segment partition $\mathbf{i}$ and, in particular, for the partition that maximizes $\mathcal{W}_r(\mathbf{i}, \psi, M)$. Using Definition (5), we obtain $IO_W(\phi, M) \geq \mathcal{W}_r(\psi, M)$. To formulate a lower bound that holds for any computation ϕ, hence for any visit ψ, we need to minimize with respect to $\psi \in \Psi_r(G_R)$, arriving at Bound (7). Bounds (8) and (9) are established by analogous arguments. ◀

5.3 Comparison with Hong and Kung's S-partition technique

Hong and Kung [23] introduced the "*S-partition technique*", for I/O lower bounds. In this section, we show that the central result of their approach can be derived as a corollary of the visit partition approach, when the latter is specialized to the topological visit rule $r^{(top)}$.

The S-partitions of a DAG are defined in terms of dominator and minimum sets. Given a DAG $G = (V, E)$ and a set $V' \subseteq V$, we say that $D \subseteq V$ is a *dominator set* of V' if every directed path from an input vertex of G to a vertex in V' intersects D. The *minimum set* of V' is the set of all vertices of V' that have no successors in V'. An S-partition is a sequence $(V_1, V_2, \ldots, V_k)$ of sets such that (a) they are disjoint and their union equals V; (b) each V_j has a dominator set of size at most S; (c) the minimum set of each V_j has size at most S; (d) there is no edge from a vertex in V_j to a vertex in $\cup_{i=1}^{j-1} V_i$.

▶ **Theorem 16** (Adapted from [23, Theorem 3.1])**.** *Any computation of a DAG $G = (V, E)$ on the I/O model with a cache of M words, executing q I/O operations, is associated with a $2M$-partition of G with k sets, such that $Mk > q > M(k-1)$. Therefore, if $k(G, 2M)$ is the minimum size of a $2M$-partition of G, the I/O complexity of G satisfies:*

$$IO(G, M) \geq M\,(k(G, 2M) - 1)\,. \tag{10}$$

Next, we show that when $r = r^{(top)}$, Bound (9) of Theorem 15 implies Bound (10) of Theorem 16. In the visit framework, minimum sets arise as the boundary of $r^{(top)}$-visits.

▶ **Theorem 17** (Comparison with S-partition technique)**.** *Given a DAG G, let ψ be any $r^{(top)}$-visit of G_R. There exists at least one segment partition $\mathbf{i} = (i_1, \ldots, i_k)$ of ψ such that*

$$\mathcal{W}_r(\mathbf{i}, \psi, M) + \mathcal{R}_r(\mathbf{i}, \psi, M) \geq M\,(k(G, 2M) - 1)\,,$$

whence $IO(G, M) \geq M\,(k(G, 2M) - 1)$.

The proof of Theorem 17 is presented in the Appendix.

Diamond DAGs can be obtained by taking a $b \times b$ mesh (*i.e.,* a two-dimensional array) and by directing all the edges towards the upper right corner. The graph obtained as such has $n = b^2$ vertices, a single input vertex (*i.e.,* in the bottom left corner), and a single output vertex (*i.e.,* in the upper right corner). We show that for these DAGs, the visit partition technique yields tight I/O lower bounds, whereas the S-partition technique would only yield an $\Omega(1)$ lower bound.

Let $G = (V, E)$ be a b-side Diamond DAG. According to the definition of S-partition, the family $\{V\}$ is a 1-partition of V as V has a dominator (resp., minimum set) of size 1 composed by the single input (resp., output) vertex. Hence, for any $M \geq 1$, $k(G, 2M) = 1$ so that Hong and Kung's method in Theorem 16 ([23 , Theorem 3.1]) yields a trivial bound. In contrast, the visit partitioning technique allows to obtain an asymptotically tight I/O lower bound:

▶ **Theorem 18** (I/O complexity Diamond DAG). *Let $G = (V, E)$ be a diamond DAG of side b. The I/O-complexity of G using a cache memory of size M satisfies $IO(G, M) \geq b/4M$.*

The proof of Theorem 18, an asymptotically matching upper bound, and an extension to a more general family of diamond DAGs are presented in the extended version [9, Section 5.4].

5.4 Extensions to related I/O models

External Memory Model. The result in Theorem 15 can be straightforwardly extended to the External Memory Model of Aggarwal and Vitter [1], where a single I/O operation can move $L \geq 1$ memory words between cache and consecutive slow-memory locations.

Models with asymmetric cost of read and write I/O operations. Since our method distinguishes the contribution of write and reads I/O operations, it would be interesting to use it to investigate I/O lower bounds where reads and writes have different cost [12, 20], including the case in which only the cost of write I/O operations is considered [2, 14]. To this end, the lower bound in (9) can be modified to include multiplicative scaling for each component.

Dropping the slow memory requirement for output values. By using a modified concept of r-entering boundary of a segment, defined as $\hat{B}_r^{(ent)}(\psi(i_{j-1}..i_j]) = B_r(\psi[1..i_{j-1}]) \cap \psi(i_{j-1}..i_j]$, our method yields I/O lower bounds in a modified version of the I/O models where output values are not required to be written into the slow memory.

Free-input model. The visit partition technique can also be adapted to the "*free input*" model, more akin to the pebbling model, in which input values can be generated into cache at any time (e.g., they are read from a dedicated ROM memory), rather than being initially stored in the slow memory. While our lower bound to the number of write I/O operations (7) remains unchanged, the lower bound to the number of read I/O operations (8) must be revised removing the contribution of the post-dominator bound and the read I/O term of the boundary-bound. For any $r \in \mathcal{R}(G_R)$ we have:

$$IO^{fi}_{\mathcal{R}}(G, M) \geq \min_{\psi \in \Psi_r(G_R)} \mathcal{W}_r(\psi, M) - |O|,$$

and, thus, $IO^{fi}(G, M) \geq \min_{\psi \in \Psi_r(G_R)} 2\mathcal{W}_r(\psi, M) - |O|$.

Execution with no recomputation. Finally, our result in Theorem 15 can be adapted and simplified to yield I/O lower bounds assuming that the value associated to any vertex $V \setminus I$ is computed exactly once (the no-recomputation assumption). This simplifying assumption is often of interest as it focuses the analysis on schedules with a minimum number of computational steps. Moreover, it may provide a stepping stone towards the analysis of the more general and challenging case where recomputation is allowed. Without recomputation, computational schedules correspond to the topological orderings of G. Thus, for any $r \in \mathcal{R}(G_R)$, the corresponding r-visits of G_R constructed according to the procedure discussed in Section 5.2, are the topological orderings of G_R. Therefore, we can restrict our attention to such orderings obtaining the following corollary:

▶ **Corollary 19.** *Given a DAG $G = (V, E)$, with input set I and output set O, consider its computations in the I/O model using a cache of size M such that no value is ever computed more than once. For any $r \in \mathcal{R}(G_R)$ we have:*

$$\begin{aligned}
IO^{nr}_{\mathcal{W}}(G, M) &\geq \min_{\psi \in \Psi_{r^{(top)}}(G_R)} \mathcal{W}_r(\psi, M),\\
IO^{nr}_{\mathcal{R}}(G, M) &\geq \min_{\psi \in \Psi_{r^{(top)}}(G_R)} \max\{\mathcal{R}_r(\psi, M),\ \mathcal{W}_r(\psi, M) + |I| - |O|\},\\
IO^{nr}(G, M) &\geq \min_{\psi \in \Psi_{r^{(top)}}(G_R)} \{\mathcal{W}_r(\psi, M) + \max\{\mathcal{R}_r(\psi, M), \mathcal{W}_r(\psi, M) + |I| - |O|\}\}.
\end{aligned}$$

The bounds obtained for computations without recomputation are generally higher than the general ones in Theorem 15, as while for each rule r we still analyze the entering boundary and the minimum post-dominator size of visit partitions, the set of visits to be considered is restricted to the subset $\Psi_{r^{(top)}}(G_R) \subseteq \Psi_r(G_R)$, thus possibly eliminating some visit ψ with low $\mathcal{W}_r(\psi', M)$ and/or $\mathcal{R}_r(\psi', M)$. It is easy to see that the best lower bounds are obtained for the choice $r = r^{(sin)}$.

6 Conclusions

We have proposed the visit framework to investigate both space and I/O complexity lower bounds. The universal upper bounds we have obtained for both the singleton and the topological types of visits show that these types cannot yield tight pebbling lower bounds for all DAGs, although they do for some DAGs. The framework gives ample flexibility to tailor the type of visit to the given DAG, but we do not yet have good insights either on how to exploit this flexibility or on how to show that this flexibility is not ultimately helpful.

The spectrum of visit types exhibits the following tradeoff. As we go from the topological rule to, say, the singleton rule, by relaxing the enablement constraints, the set of vertex sequences that qualifies as a visit increases (which goes in the direction of reducing the boundary complexity of the DAG, since the minimization takes place over a larger domain), but the boundary complexity of a specific visit also increases (which goes in the direction of increasing the boundary complexity of the DAG, since the function to be minimized increases). The tension between these opposite forces has proven difficult to analyze quantitatively. The arguments used to establish universal upper bounds in the singleton and in the topological cases are significantly different, and it is not clear how to interpolate them for an intermediate visit type. Further research is clearly needed to make progress on what appears to be a rich combinatorial problem.

Another contribution of the visit framework is a step toward a unified treatment of pebbling and I/O complexity. Within the framework, we have already seen how to generalize the by now classical Hong-Kung partition technique, based on dominator and minimum sets, thus achieving much better lower bounds for some DAGs.

We conjecture that, for other significant DAGs whose I/O complexity cannot be well captured by the S-partition technique, visit partitions will help obtain good lower bounds. Good candidates are DAGs with a constant degree and a space complexity $S(N)$ superlinear, say polynomial, in the number of inputs $N = |I|$. For such DAGs, the size of the minimum dominator set cannot exceed N and, as shown in Theorem 17, the size of the topological boundary, *i.e.,* of the minimum set, is $O(\log N)$ at any point in the computation. On the other hand, the singleton boundary could be significantly higher.

Although we have not explicitly discussed the issue in this work, we do not have general I/O upper bounds matching our visit partition lower bounds. Therefore, further work is needed to achieve a full characterization of the I/O complexity of a DAG.

References

1 Alok Aggarwal and S. Vitter, Jeffrey. The Input/Output Complexity of Sorting and Related Problems. *Communications of the ACM*, 31(9):1116–1127, September 1988. doi:10.1145/48529.48535.

2 Grey Ballard, Dulceneia Becker, James Demmel, Jack Dongarra, Alex Druinsky, Inon Peled, Oded Schwartz, Sivan Toledo, and Ichitaro Yamazaki. Communication-avoiding symmetric-indefinite factorization. *SIAM Journal on Matrix Analysis and Applications*, 35(4):1364–1406, 2014.

3 Grey Ballard, James Demmel, Olga Holtz, Benjamin Lipshitz, and Oded Schwartz. Brief announcement: Strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds. In *Proceedings of the twenty-fourth annual ACM symposium on Parallelism in algorithms and architectures*, pages 77–79. ACM, 2012.

4 Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Communication-optimal parallel and sequential Cholesky decomposition. *SIAM Journal on Scientific Computing*, 32(6):3495–3523, 2010.

5 Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Minimizing communication in numerical linear algebra. *SIAM Journal on Matrix Analysis and Applications*, 32(3):866–901, 2011.

6 Grey Ballard, James Demmel, Olga Holtz, and Oded Schwartz. Graph expansion and communication costs of fast matrix multiplication. *Journal of the ACM (JACM)*, 59(6):1–23, 2013.

7 Gianfranco Bilardi and Lorenzo De Stefani. The I/O complexity of Strassen's matrix multiplication with recomputation. In *Workshop on Algorithms and Data Structures*, pages 181–192. Springer, 2017.

8 Gianfranco Bilardi and Lorenzo De Stefani. The I/O Complexity of Toom-Cook Integer Multiplication. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 2034–2052, USA, 2019. Society for Industrial and Applied Mathematics.

9 Gianfranco Bilardi and Lorenzo De Stefani. The dag visit approach for pebbling and i/o lower bounds, 2022. doi:10.48550/arXiv.2210.01897.

10 Gianfranco Bilardi, Andrea Pietracaprina, and Paolo D'Alberto. On the space and access complexity of computation DAGs. In *Graph-Theoretic Concepts in Computer Science*, pages 47–58. Springer, 2000.

11 Gianfranco Bilardi and Franco Preparata. Processor-Time trade offs under bounded speed message propagation. Part 2: Lower Bounds. *Theory of Computing Systems*, 32(5):531–559, 1999.

12 Guy E. Blelloch, Jeremy T. Fineman, Phillip B. Gibbons, Yan Gu, and Julian Shun. Efficient Algorithms with Asymmetric Read and Write Costs. In Piotr Sankowski and Christos D. Zaroliagis, editors, *24th Annual European Symposium on Algorithms, ESA 2016, August 22-24, 2016, Aarhus, Denmark*, volume 57 of *LIPIcs*, pages 14:1–14:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ESA.2016.14.

13 Timothy Carpenter, Fabrice Rastello, P. Sadayappan, and Anastasios Sidiropoulos. Brief Announcement: Approximating the I/O Complexity of One-Shot Red-Blue Pebbling. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '16, pages 161–163, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2935764.2935807.

14 Erin C. Carson, James Demmel, Laura Grigori, Nicholas Knight, Penporn Koanantakool, Oded Schwartz, and Harsha Vardhan Simhadri. Write-Avoiding Algorithms. In *2016 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2016, Chicago, IL, USA, May 23-27, 2016*, pages 648–658. IEEE Computer Society, 2016. doi:10.1109/IPDPS.2016.114.

15 Lorenzo De Stefani. *On space constrained computations*. PhD thesis, University of Padova, 2016.

16 Lorenzo De Stefani. Brief Announcement: On the I/O Complexity of Sequential and Parallel Hybrid Integer Multiplication Algorithms. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '22, pages 449–452, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3490148.3538551.

17 Venmugil Elango, Fabrice Rastello, Louis-Noël Pouchet, Jagannathan Ramanujam, and Ponnuswamy Sadayappan. On characterizing the data access complexity of programs. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 567–580, 2015.

18 Harvey Friedman. Algorithmic procedures, generalized turing algorithms, and elementary recursion theory. In *Studies in Logic and the Foundations of Mathematics*, volume 61, pages 361–389. Elsevier, 1971.

19 D. Yu Grigor'ev. Application of separability and independence notions for proving lower bounds of circuit complexity. *Zapiski Nauchnykh Seminarov POMI*, 60:38–48, 1976.

20 Yan Gu, Yihan Sun, and Guy E. Blelloch. Algorithmic building blocks for asymmetric memories. In Yossi Azar, Hannah Bast, and Grzegorz Herman, editors, *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, volume 112 of *LIPIcs*, pages 44:1–44:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ESA.2018.44.

21 John Hopcroft, Wolfgang Paul, and Leslie Valiant. On time versus space. *Journal of the ACM (JACM)*, 24(2):332–337, 1977.

22 Dror Irony, Sivan Toledo, and Alexander Tiskin. Communication lower bounds for distributed-memory matrix multiplication. *Journal of Parallel and Distributed Computing*, 64(9):1017–1026, 2004.

23 Hong Jia-Wei and Hsiang-Tsung Kung. I/O complexity: The red-blue pebble game. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing*, pages 326–333, 1981.

24 Lynn H Loomis and Hassler Whitney. An inequality related to the isoperimetric inequality. *Bulletin of the American Mathematical Society*, 55(10):961–962, 1949.

25 Auguste Olivry, Guillaume Iooss, Nicolas Tollenaere, Atanas Rountev, P. Sadayappan, and Fabrice Rastello. Ioopt: automatic derivation of I/O complexity bounds for affine programs. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pages 1187–1202. ACM, 2021. doi:10.1145/3453483.3454103.

26 Rasmus Pagh and Morten Stöckel. The input/output complexity of sparse matrix multiplication. In *European Symposium on Algorithms*, pages 750–761. Springer, 2014.

27 Michael S Paterson and Carl E Hewitt. Comparative schematology. In *Record of the Project MAC conference on concurrent systems and parallel computation*, pages 119–127. ACM, 1970.

28 Wolfgang J Paul, Robert Endre Tarjan, and James R Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10(1):239–251, 1976.

29 Desh Ranjan, John Savage, and Mohammad Zubair. Upper and lower I/O bounds for pebbling r-pyramids. *Journal of Discrete Algorithms*, 14:2–12, 2012.

30 J. E. Savage. Extending the Hong-Kung model to memory hierarchies. In *Computing and Combinatorics*, pages 270–281. Springer, 1995.

31 John E. Savage. *Models of Computation: Exploring the Power of Computing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1997.

32 Jacob Scott, Olga Holtz, and Oded Schwartz. Matrix multiplication I/O-complexity by path routing. In *Proceedings of the 27th ACM symposium on Parallelism in Algorithms and Architectures*, pages 35–45, 2015.

33 Michele Scquizzato and Francesco Silvestri. Communication lower bounds for distributed-memory computations. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 627–638. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.627.

A Proofs of technical results

Proof of Theorem 5. Consider a pebbling schedule ϕ of G which uses s pebbles, and any visit rule $r \in \mathcal{R}(G_R)$. The proof proceeds by constructing an r-visit ψ of G_R whose r-boundary complexity is itself bounded from above by s. Let T denote the total number of steps of the pebbling ϕ. The construction of ψ proceeds iteratively starting from the end of the pebbling schedule ϕ to its beginning: Let v denote the vertex which is being pebbled at the i-th step of ϕ for $1 \leq i \leq T$, then the same vertex v is visited in G_R if and only if all the vertices in at least one of the subsets in the enabling family $h(v)$ have already been visited. By construction, ψ is indeed a valid r-visit of G_R.

By the construction of the reverse DAG G_R, the set of the successors of any vertex $v \in V$ in G corresponds to the set of predecessors of v in G_R. By the rules of the pebble game, when considering a complete pebbling schedule for G, each vertex $v \in V$ is pebbled in for the first time before any of its successors. As each enabler in $r(v)$ is a subset of the predecessors of v in G_R and, hence, a subset of the successors of v in G, each vertex will surely be visited in ψ at the step corresponding to its first pebbling in ϕ unless it has already been visited. This allows us to conclude that all vertices of G_R are indeed visited by ψ.

Let $\phi[t]$ denote the vertex being pebbled at the t-th step of the pebbling schedule, for $1 \leq t \leq T$. In order to prove that the statement holds, it must be shown that, fixed an index i, $1 \leq i \leq n$ with $\psi[i] = \phi[t]$, for some $t \in \{1, 2, \ldots, T\}$, the vertices in $B_r(\psi[1..i])$ must be pebbled (*i.e.,* held in the memory) at the end of t-th step of ϕ.

Let $v \in B_r(\psi[1..i])$. By the construction of ψ there must exist two indices t_1 and t_2, with $1 \leq t_1 \leq t < t_2 \leq T$, such that $\phi[t_1] = v$, $\phi[t_2] \in Q \in r(v)$, and $\phi[t'] \neq v$ for every $t_1 < t' < t_2$ (if that was not the case, then v would have been visited then). As a consequence, the value of v computed at step t_1 of ϕ is used to compute v_{t^2} and therefore it must reside in memory at the end of step t (*i.e.,* it has to be pebbled). Since i was chosen arbitrarily, the same reasoning applies for all indices $i = 1, 2, \ldots, n$. We can thus conclude that the maximum number of pebbles used by ϕ (and thus, the memory space used by ψ) is no less than $\max_{1 \leq i \leq n} |B_r(\psi[1..i])| = b_r(\psi)$.

The theorem follows by minimizing over all possible r-visits $\psi \in \Psi_r(G_R)$. ◀

Proof of Lemma 6. To prove the lemma, we construct a pebbling schedule ϕ for G, which corresponds to the r-visit ψ of G_R: the construction proceeds iteratively starting from the end of ψ. For any index $i = 1, 2, \ldots, n$ let $\psi[i]$ denote the vertex visited at the i-th step of ψ.

Let G_n denote the sub-DAG of G induced by the subset of V composed of $\psi[n]$ and the set of its ancestors in G. The schedule ϕ starts following the steps of any pebbling schedule of G_n. Once $\psi[n]$ has been pebbled, all the pebbles on vertices of G_n, except for $\psi[n]$, are removed. The schedule ϕ then proceeds according to the same procedure up to $\psi[1]$. During the i-th step, the pebbling schedule for G_i never re-pebbles any of the vertices in $\psi[i+1..n]$. As they were previously pebbled, we can assume that they maintain a pebble (*i.e.,* they are kept in memory) until the end of the computation. Hence it is possible to pebble G_i without re-pebbling the vertices in $\psi[i+1..n]$. By construction, ϕ is a complete pebbling of G.

Crucially, the order according to which the vertices are pebbled for the *last time* in ϕ corresponds to the reverse order of appearance of the vertices in the given visit ψ. When applying the conversion between pebbling schedules and visits discussed in the proof of Theorem 5, we have that the vertices are visited in G_R according to the order of their *last pebbling* in ϕ. The lemma follows. ◀

Proof of Theorem 9. We construct inductively an r-visit ψ of G such that $b_r(\psi) \leq \ell\,(d_{out}-1)$.

In the base case $\ell = 0$, that is, all the vertices of G are input vertices without predecessors. Any permutation of the input vertices is an r-visit. As input vertices are enabled by the empty set, by Definition 3, they do not appear in the boundary and, thus, the r-boundary complexity of any such visit is zero.

For the general case $\ell \geq 1$, the visit begins by visiting any input vertex of G. If none of its direct successors are enabled according to r, by definition, $B_r(\psi[1]) = \emptyset$. If that is the case, the visit proceeds by selecting another input vertex of G.

Without loss of generality, let v denote the first input vertex of G whose r-enabled reach given the r-sequence ψ_{pre} constructed so far is not empty. As visiting v can only enable its at most d_{out} successors, we have $|B_r(\psi_{pre})| \leq d_{out}$. Let $G' = (V', E')$ denote the sub-DAG of G induced by the r-enabled reach of v given ψ_{pre} and let $r'(v) = \{Q \setminus \psi_{pre} \mid Q \in r(v)\}$ for all $v \in \mathrm{reach}_r(v|\psi_{pre})$. By construction, G' is a sub-DAG of the DAG induced by the set of descendants of v, whose topological depth must be at most $\ell - 1$. Thus, G' has topological depth at most $\ell - 1$ as well. Hence, by the inductive hypothesis, for any visit rule of G', and, in particular, for r' there exists an r'-visit of G', denoted as ψ' such that $b_{r'}(\psi') \leq (\ell-1)(d_{out}-1)+1$.

By Lemma 8, $\psi_{pre}\psi'$ is an r-sequence of G and vertices in ψ' do not enable any vertex in $V \setminus \psi'$. Further, as at the first step of ψ' one successor of v is visited, from that step onward, at most $d_{out}-1$ successors of v which are yet to be visited may be in the boundary. Thus:

$$
\begin{aligned}
|B_r(\psi_{pre}[1..i])| &= 0 && \text{for } i = 1, \ldots, |\psi_{pre}| \\
|B_r(\psi_{pre})| &\leq d_{out} \\
|B_r(\psi_{pre}\psi'[1..j])| &\leq (d_{out}-1) + (\ell-1)(d_{out}-1) + 1 && \text{for } j = 1, \ldots, |\psi'| \\
|B_r(\psi_{pre}\psi')| &\leq (d_{out}-1)
\end{aligned}
$$

The visit then proceeds by visiting any input vertex of G which is yet to be visited and its enabled reach given the r-sequence constructed so far and by repeating the operations previously described. This ensures that G is entirely visited by ψ. By repeating the considerations on the boundary size previously discussed, we can conclude that the maximum boundary size of ψ is at most $(d_{out}-1)\,\ell + 1$. The theorem follows. ◀

Proof of 12.

(a) If $d_{out} = 0$, then all vertices of G are input vertices and any ordering ψ of V is a legal topological ordering, with $b_{r(top)}(\psi) = 0$, since input vertices are never part of the boundary.

(b) If $d_{out} = 1$, then it is easy to see that G consists of a collection of chains (one for each input), at least one of which has length greater than 1. Clearly, visiting one chain at the time yields a visit ψ with $b_{r(top)}(\psi) = 1$.

(c) If $d_{out} \leq D$, with $D \geq 2$, we proceed by induction on n. In the base case, $n = 1$, clearly there is a unique schedule ψ with $b_{r(top)}(\psi) = 0$. For the inductive step ($n > 1$), we recursively construct a visit $\psi = topv(G)$ as follows. Arbitrarily choose an input vertex u of G; let $B_{r(top)}(u) = \{v_1, v_2, \ldots, v_k\} \subseteq \mathrm{suc}(u)$, with $0 \leq k \leq d_{out}(u) \leq D$; and output $\psi = uv_{j_1}\phi_1 \ldots v_{j_k}\phi_k\phi$ where, informally, ϕ_h is a visit of the enabled reach of v_{j_h}, given the prefix of ψ to the left of v_{j_h}. Each v_{j_h} is chosen among the successors of u (initially) enabled by u and not yet visited so as to minimize the size of the enabled reach. Finally, ϕ is a visit of the vertices not yet traversed by the end of ϕ_k. More formally:

- $j_1, \ldots, j_k$ is a permutation of $1, \ldots, k$, specified below.
- For $h = 1, \ldots, k$, $v_{j_h}\phi_h = topv(G_h)$, where G_h is the subgraph induced by $\{v_{j_h}\} \cup \mathrm{reach}_{r(top)}(v_{j_h}|uv_{j_1}\phi_1 \ldots v_{j_{h-1}}\phi_{h-1}v_{j_h})$. It is easy to see that any visit of G_h must begin with vertex v_{j_h}, the only input vertex of G_h, every other vertex of G_h being a proper descendant of v_{j_h}.
- $j_h = \underset{j \in \{1,\ldots,k\} \setminus \{j_1,\ldots,j_{h-1}\}}{\arg\min} |\mathrm{reach}_{r(top)}(v_j|uv_{j_1}\phi_1 \ldots v_{j_{h-1}}\phi_{h-1}v_j)|$.
- $\phi = topv(\bar{G})$, where $\bar{G}$ is the subgraph induced by $V \setminus \{uv_{j_1}\phi_1 \ldots v_{j_k}\phi_k\}$.

To establish the claimed bound on $b_{r(top)}(\psi)$, we make the following observations.

1. For $h = 1, \ldots, k$, the boundary of the prefix of ψ ending just before v_{j_h} equals $\{v_{j_h}, \ldots, v_{j_k}\}$, hence it has size $k - h + 1 \leq k$.
2. By Lemma 8, while visiting G_h, the boundary is at most $(k - h) + b_{r(top)}(v_{j_h}\phi_h)$.
3. Collectively, the $k - h + 1$ enabled reaches from which the next one to be visited is chosen contain at most $n - 1 - k$ vertices (because u and $v_1, \ldots, v_k$ are not contained in any of them). Since, by Lemma 11, these reaches are disjoint, the smallest ones contain at most $\lfloor \frac{n-1-k}{k-h+1} \rfloor$ vertices. Additionally accounting for v_{j_h}, G_h contains at most $1 + \lfloor \frac{n-1-k}{k-h+1} \rfloor = \lfloor \frac{n-h}{k-h+1} \rfloor$ vertices.
4. For convenience, let $f(n, D) = \frac{D-1}{\log_2 D} \log_2 n + 1$ and observe that f is increasing with both arguments. By the inductive hypothesis, we have:
$$b_{r(top)}(\psi) \leq \max\{k, \max_{h=1}^{k}\{(k - h) + f(\frac{n - h}{k - h + 1}, D)\}, f(n - 1 - k, D)\}.$$
Using $f(\frac{n-h}{k-h+1}, D) \leq f(\frac{n}{k-h+1}, D)$ and letting $q = k - h + 1$, the previous yields
$$b_{r(top)}(\psi) \leq \max\{k, \max_{q=1}^{k}\{q - 1 + f(\frac{n}{q}, D)\}, f(n - 1 - k, D)\}.$$
5. To complete the inductive step, it remains to show that each of the three terms in the outer max is no larger than $f(n, D)$. This is obvious for the third term $f(n - 1 - k, D)$, given the monotonicity of f. For the second term, we observe that each argument in the inner max satisfies
$$\begin{aligned} q - 1 + f(\frac{n}{q}, D) &= q - 1 + \frac{D-1}{\log_2 D} \log_2(\frac{n}{q}) + 1 \\ &= \left(\frac{D-1}{\log_2 D} \log_2 n + 1\right) + (q - 1)\left(1 - \frac{D-1}{\log_2 D}\frac{\log_2 q}{q - 1}\right) \\ &\leq \frac{D-1}{\log_2 D} \log_2 n + 1, \end{aligned}$$
where the term dropped in the last step is negative or null, since $\frac{q-1}{\log_2 q} \leq \frac{D-1}{\log_2 D}$, for $q = 1, \ldots, D$, as can be shown by straightforward calculus. Finally, to bound the first term, k, we observe that, if $k \leq 1$, then the target bound trivially holds. Otherwise,

consider that $n \geq k+1$ (G contains at least the distinct vertices u and $v_1, \ldots, v_k$) and that, for $2 \leq k \leq D$, we have $\frac{k-1}{\log_2 k} \leq \frac{D-1}{\log_2 D}$ (as mentioned above). Then, the target inequality follows:

$$k = (k-1) + 1 \leq \frac{k-1}{\log_2 k} \log_2 k + 1 < \frac{D-1}{\log_2 D} \log_2 n + 1. \qquad ◀$$

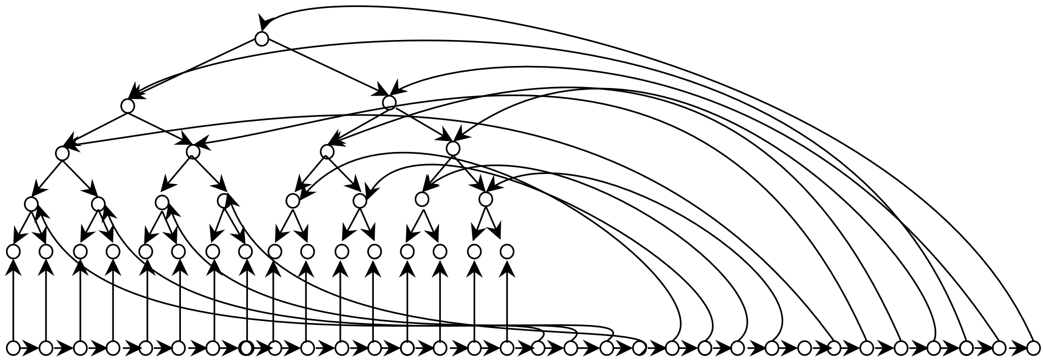

Figure 1 This DAG allows us to study the relationship between the $r^{(sin)}$-boundary complexity and the $r^{(top)}$-boundary complexity. The DAG is obtained by connecting the vertices of a directed binary arborescence with a directed chain. Its $r^{(sin)}$-boundary complexity is 2: consider a visit starting in the leftmost vertex of the chain and that visits vertices of the tree as soon as they are enabled. Instead, its $r^{(top)}$-boundary complexity is $\log_2 (n+2) - 1$, where n denotes the number of its vertices. In every possible $r^{(top)}$-visit, the directed chain must be visited first, and then the arborescence is left to be visited. The bound on the $r^{(top)}$- boundary complexity follows an argument similar to that in the analysis of complete q-trees in the extended version of this work [9]. Consider the reverse DAG. Its pebbling number is $(n+2)-1$, where n denotes the number of vertices. (The proof of this statement is a simple exercise.) By the previous considerations, while using the $r^{(sin)}$ yields in Theorem 5 yields a trivial lower bound to the pebbling number. Instead, using $r^{(top)}$ yields a tight lower bound to the pebbling number of the DAG.

Proof of Theorem 17. For notational simplicity, throughout this proof, r stands for $r^{(top)}$. We preliminarily observe that if $\psi \in \Psi_{r^{(top)}}(G_R)$, then, for any segment $\psi(h,i]$, $B_r^{(ent)}(\psi(h,i])$ is the minimum set of $\psi(h,i]$ in G. In fact, by the definition of $r^{(top)}$, $v \in B_r(\psi[1,h])$ if and only if *all* of its predecessors in G_R (*i.e.*, its successors in G) are in $\psi[1..h]$. Hence, $B_r^{(ent)}(\psi(h..i])$ contains exactly those vertices in $\psi(h,i]$ which are either inputs of G_R (outputs of G) or for which $\psi(h,i]$ contains no predecessor in G_R (thus, no successor in G).

Below we will make use of the following two properties, whose simple proof is omitted here. Let $U \subseteq V$ and $x \in V$. Then $min_{pd}(U \cup \{x\}) \leq min_{pd}(U)+1$ and $sms(U \cup \{x\}) \leq sms(U)+1$, where $sms(X)$ denotes the size of the minimum set of $X \subseteq V$.

Given $\psi \in \Psi_{r^{(top)}}(G_R)$, we consider a segment partition $\mathbf{i} = (i_1, i_2, \ldots, i_k)$ where each segment, with the possible exception of the last one, has a minimum postdominator or the minimum set of size $2M$. Based on the properties stated in the preceding paragraph, such a partition can be easily constructed by scanning the visit one vertex at a time and closing a segment as soon as the desired condition is met, or all vertices have been scanned.

For $j = 1, \ldots, k$, let $V_j = \{\psi[i_{k-j}+1], \ldots, \psi[i_{k+1-j}]\}$. We show that sequence $(V_1, \ldots, V_k)$ is a $2M$-partition of G, by proving the defining properties: (a) As they correspond to the segments of a visit, the V_j's are disjoint and their union equals V. (b) By construction, the dominator of each V_j in G has size at most $2M$. (c) By construction, the minimum set of each V_j in G has size at most $2M$. (d) Finally, since ψ is an $r^{(top)}$-visit of G_R, it is a reverse topological ordering of the vertices in G. Therefore, there are no edges of G from V_j to $\cup_{i=1}^{j-1} V_i$. Clearly, $k \geq k(G, 2M)$, by the definition of the latter quantity.

To analyze the I/O requirements of ψ, consider the following sequence of inequalities:

$$\begin{aligned}
\mathcal{W}_r(\psi, M) \quad + \quad & \mathcal{R}_r(\psi, M) \geq \mathcal{W}_r(\mathbf{i}, \psi, M) + \mathcal{R}_r(\mathbf{i}, \psi, M) \\
\geq \quad & \sum_{j=1}^{k} \max\{0, b_r^{(ent)}\left(\psi(i_{j-1}..i_{j+1}]\right) - M, min_{pd}\left(\psi(i_{j-1}..i_j]\right) - M\} \\
\geq \quad & \sum_{j=1}^{k-1} (2M - M) = (k-1)M \\
\geq \quad & M\left(k(G, 2M) - 1\right).
\end{aligned}$$

These four inequalities respectively take into account (i) the definitions in (5) and (6); (ii) the definitions in (1) and (3); (iii) the fact that, for $j = 1, \ldots, k-1$, at least one of the arguments of the max operator equals M; and (iv) the relationship $k \geq k(G, 2M)$, seen above. Finally, since the chain of inequalities applies to any visit $\psi \in \Psi_{r^{(top)}}\left(G_R\right)$, we can invoke inequality (9) to conclude that $IO\left(G, M\right) \geq M\left(k(G, 2M) - 1\right)$. ◀