On The Space and Access Complexity of Computation DAGs*

Gianfranco Bilardi[1], Andrea Pietracaprina[1], and Paolo D'Alberto[2]

[1] Dipartimento di Elettronica e Informatica, Università di Padova, Italy.
{bilardi,andrea}@art.dei.unipd.it
[2] Information and Computer Science, University of California at Irvine
paolo@ics.uci.edu

Abstract. We study the space and the access complexity of computations represented by Computational Directed Acyclic Graphs (CDAGs) in hierarchical memory systems. First, we present a unifying framework for proving lower bounds on the space complexity, which captures most of the bounds known in the literature for relevant CDAGs, previously proved through *ad-hoc* arguments. Then, we expose a close relationship between the notions of space and access complexity, where the latter represents the minimum number of accesses performed by any computation of a CDAG at a given level of the memory hierarchy. Specifically, we present two general techniques to derive bounds on the access complexity of a CDAG based on the space complexity of certain subgraphs. One technique, simpler to apply, provides only lower bounds, while the other provides (almost) matching lower and upper bounds and improves upon previous well-known result by Hong and Kung.

1 Introduction

The substantial fraction of the cost of a computing system accounted for by memory has motivated the study of space-efficient computations since the early times of automatic computing. However, a more fundamental reason to strive for space efficiency arises directly from the pursuit of performance: due to basic physical principles, access time is bound to increase with the size of memory. Ultimately, smaller memory translates into faster computation, making space efficiency a crucial objective, even when the cost of memory is negligible.

When a large overall memory is required, it becomes convenient to organize it hierarchically, into a sequence of levels whose size and access time increase progressively. Currently, the levels of the memory hierarchy include the register file, two or three caches, main memory, and disks. Compared to registers, main memory is a few hundred times slower and disks are a few million times slower. Hence, effective use of the faster levels of the memory hierarchy is becoming a

* This work was supported, in part, by CNR and MURST of Italy. Part of the work of G. Bilardi was done while this author was a visitor at the IBM T. J. Watson Research Laboratory.

key concern in the design of algorithms. As in all endevours where performance is systematically pursued, it is important to be able to evaluate the distance from optimality of a proposed solution. The present paper focuses on the development of general techniques that can be employed to prove bounds on certain relevant measures related to the usage of the memory hierarchy by a computation.

We consider the problem of finding an optimal *implementation* of a computation which has been specified in terms of a *Computational Directed Acyclic Graph* (CDAG), whose vertices represent operations (of both input and processing type) and whose arcs represent data dependencies. The degrees of freedom of the implementor are essentially the *schedule* of execution of operations, possibly including repetitions, and the *memory management*, that is, the assignment of a memory location to each value arising in the computation during the time between production and last use of such value. An extensive literature on the space complexity of CDAGs, typically formulated in terms of so-called *pebbling games*, has developed in the past (see e.g., [PH70,Coo74,HPV77,Pip80,LT82,PTC77,RS83]). A pioneering paper by Hong and Kung [HK81] begun exploring complexity in two-level memory hierarchies and proposed a technique to bound from below the number of accesses performed at the slowest level of the hierarchy for a given CDAG. To date, this paper remains the main point of departure of most lower bound analysis for hierarchical memory performance (see, e.g., [AACS87,AV88]). Savage [Sav93] developed an interesting reformulation of the Hong and Kung's technique as well as an extension to multilevel hierarchies. (A comprehensive survey of all of the above results can be found in [Sav98, Ch.10,11].) In a recent work, Bilardi and Preparata [BP99] proposed a new lower bound technique based on the notion of *dicothomy-width*, which is restricted to implementations where every operation is executed exactly once. Such technique has been successfully extended to distributed memory hierarchies.

In this paper, we study both the space and the access complexity of CDAGs, and, in particular, expose a tight relationship between the two. Models and relevant quantities are defined in Section 2. In Section 3, we introduce a family of graph-theoretic measures that can be employed to prove lower bounds to the space complexity of CDAGs. Each measure is associated with a suitable class of permutations of the vertices, and it is defined in terms of a suitable notion of boundary size for the prefixes of such permutations. More specifically, the measure is the minimum, over all the permutations in the class, of the maximum boundary size of a prefix. The key link between the space complexity of a CDAG and each measure is established by showing that each execution of the CDAG must embed a permutation of the operations belonging to the associated class and that, at each stage of the execution, the values of nodes in the current prefix must be stored in memory. We show how most of the results in the literature on space complexity, which are derived by *ad-hoc* arguments for the CDAG at hand, can be systematically derived in the outlined framework.

In Section 4, we turn our attention to the *access complexity* $Q_G(s)$ of a CDAG G defined as minimum number of accesses to locations with address $x \geq s$, over all executions of G, on a memory with locations addressed by the natural numbers starting from 0. First, we discuss a simple but effective technique that affords the derivation of lower bounds on $Q_G(s)$ from the space complexity of disjoint subdags of G. Second, we show, in a sense that can be made precise, that $Q_G(s)$ is closely related, by both lower and upper bounds, to the minimum number of subdags of space complexity at most $2s$ that "cover" the given CDAG. On the side of lower bounds, our results encompass those of both [HK81] and [Sav98]; on the side of upper bounds, we are not aware of any previous comparable results. In spite of their greater generality, our arguments are somewhat simpler than the previous ones. Among other things, they show that the concept of *minimum set* featuring in the analysis of Hong and Kung is not essential. In Section 5, we briefly indicate some directions for further research.

2 Preliminaries

Let $G = (V, E)$ be a CDAG where the nodes represent values produced by unit-time operations, and the arcs represent data dependencies. For every arc $(u, v) \in E$ we say that u is a *predecessor* of v and v is a *successor* of u. The set of predecessors of a node v are the operands of the operation that produces v. The *in-degree* (resp., *out-degree*) of a node is the number of its predecessors (resp., successors). Nodes of in-degree (resp., out-degree) 0 are regarded as *inputs* (resp., *outputs*) of the CDAG. A *computation* of G specifies a particular scheduling of the operations associated with its nodes, which satisfies data dependences, and a particular memory management.

We study CDAG computations on the RAM model with a memory of unbounded size whose cells are addressed by the natural numbers starting from 0. A *standard computation* of a CDAG G starts with the values of all input nodes in memory and must determine the values of the output nodes by performing a sequence of *node evaluations*, where a node evaluation performs the operation associated with a node v, provided that the operands of v are in memory, and stores the result in memory.

In general, a node can be evaluated more than once. The space needed by a computation is defined as the maximum number of values concurrently residing in memory at any one time during the computation. We define the *space complexity* of G, denoted by $S(G)$, as the minimum space over all standard computations of G.

In the paper we will also refer to a slightly different class of CDAG computations, called *free-input computations*, that adopt the following rule: the computation of a CDAG G starts with an initially empty memory and produces the value of each input node, every time it is needed, by invoking a special `load` instruction. It is easy to argue that the space complexity of the free-input computations of G, which we denote by $S_{\text{free}}(G)$, is not larger than $S(G)$, hence any lower bound to $S_{\text{free}}(G)$ applies to $S(G)$ as well. We observe that $S_{\text{free}}(G)$ is also the measure of space captured by the *Red Pebble Game* model [HPV77,Sav98].

Suppose that the memory is partitioned into two levels, namely L1 consisting of the s cells of address $x \in [0, s)$, and L2 consisting of the cells of address $x \geq s$, featuring respectively low and high access costs. For a CDAG G we define the *access complexity* $Q_G(s)$ as the minimum number of accesses to L2 performed by any computation of G. When studying the access complexity we will refer only to standard computations.

3 Space Complexity

If we rule out multiple executions of the same operation, the evaluation schedules of a CDAG are in one-to-one correspondence with the topological orderings of its nodes. Furthermore, for a given topological ordering, say $\phi = \phi_1\phi_2\cdots\phi_n$, one can easily argue that, immediately after evaluation of node ϕ_i, every $v \in \{\phi_1, \ldots, \phi_i\}$ with at least one successor in $\{\phi_{i+1}, \ldots, \phi_n\}$ must be in memory. Repetition of operations greatly complicates the analysis, because the simple and convenient correspondence between topological orderings and valid schedules no longer holds, and it is difficult to identify what must be in memory at any given point of a schedule.

The contribution of this section is a framework that associates arbitrary schedules (where repetitions are allowed) with suitable permutations of the vertices, while maintaining a grip on space requirements. Informally the approach works as follows. Consider an arbitrary n-node CDAG G. We introduce the notion of marking rule f as a criterion to associate with each node v of G a family $f(v)$ of subsets of its successors. Given a computation of G, the marking rule can be employed to "mark" exactly one evaluation for each node of G. The marked evaluations, in order of execution, define a permutation $\phi = \phi_1\phi_2\cdots\phi_n$ of the nodes of G with the following property. For $1 \leq i \leq n$, there is a time during the computation such that every ϕ_j with $j \leq i$ and with at least one of the subsets belonging to $f(\phi_j)$ included in $\{\phi_{i+1}, \ldots, \phi_n\}$ must be in memory. (Such ϕ_j's are said to form the boundary of the prefix.) We should observe that, when applied to a given CDAG G, a marking rule provides a lower bound to $S_{free}(G)$. In general, different rules generate different classes of permutations as well as different types of boundaries, often with opposite impact on the lower bound measure. The approach outlined above is formalized next and then applied to some well-known CDAGs.

3.1 A General Lower Bound

Let $G = (V, E)$ be an n-node CDAG. Let also $O \subseteq V$ denote the set of output nodes, and $N(v)$ the set of successors of v, for every $v \in V$. A *marking rule* for G is a function f which maps every $v \in V$ to a family of subsets of $N(v)$ such that if $v \in O$ then $f(v)$ contains only the empty set, while if $v \in V \backslash O$ then $f(v)$ cannot contain the empty set and must contain at least one (nonempty) subset of $N(v)$. In the paper we will often refer to a particular marking rule, the *singleton marking rule* $f_G^{(sing)}$, where for every $v \in V \backslash O$ $f_G^{(sing)}(v) = \{\{u\} \ : \ u \in N(v)\}$.

A permutation $\phi = \phi_1\phi_2\cdots\phi_n$ of the nodes of G is an *f-marking* of G *iff* for every $1 \leq i \leq n$ there exists $q \in f(\phi_i)$ such that $q \subseteq \{\phi_j \ : \ i < j \leq n\}$.

Moreover, we define the *i-boundary* of any *f-marking* ϕ as the set

$$B^f_\phi(i) = \{v \in \{\phi_1, \ldots, \phi_i\} \backslash O \ : \ \exists q \in f(v) \text{ s.t. } q \subseteq \{\phi_{i+1}, \ldots, \phi_n\}\} .$$

In other words, $B^f_\phi(i)$ represents the set of nodes $v \in V \backslash O$ such that $v \phi_{i+1} \cdots \phi_n$ is the suffix of a legal f-marking of G.

The following theorem shows a relation between the space complexity of the free-input computations of G and the size of the boundaries of its f-markings. Let F_G denote the set of marking rules for G and, for $f \in F_G$, let $\Phi(f)$ denote the set of f-markings of G.

Theorem 1. *The space complexity of the free-input computations of G satisfies:*

$$S_{\text{free}}(G) \geq \max_{f \in F_G} \ \min_{\phi \in \Phi(f)} \ \max_{1 \leq i \leq n} |B^f_\phi(i)|.$$

Proof: Consider an arbitrary function $f \in F_G$ and a T-step free-input computation C of G. Let v_t be the node evaluated at step t of C, for $1 \leq t \leq T$. We can construct an f-marking $\phi = \phi_1 \phi_2 \cdots \phi_n$ of G by sweeping C backwards through the following loop.

$j = n$;
`for` $t = T$ `down-to 1 do`
  `if` $(v_t \notin \{\phi_{j+1}, \ldots, \phi_n\})$ `and` $(\exists\, q \in f(v_t) \ : \ q \subseteq \{\phi_{j+1}, \ldots, \phi_n\})$
  `then` $\{$`set` $\phi_j = v_t$; $j = j - 1;\}$

It is easy to verify that at the end of the loop the resulting sequence ϕ forms indeed an f-marking of G. For $1 \leq j \leq n$ let $\tau(j)$ denote the index such that the node ϕ_j is set at iteration $\tau(j)$ (hence $\phi_j = v_{\tau(j)}$). For an arbitrary index i we now show that the values of the nodes in $B^f_\phi(i)$ must be in memory at the end of step $\tau(i)$ of C. Consider, in fact, a node $\phi_j \in B^f_\phi(i)$. The definition of $B^f_\phi(i)$ implies that $j \leq i$ and that $\{\phi_{i+1}, \ldots, \phi_n\}$ includes a subset $q \in f(\phi_j)$. Let $k \geq i + 1$ be the smallest index such that $\phi_k \in q$. It is easy to see that $\tau(j) \leq \tau(i) < \tau(k)$ and that for every $\tau(j) < t < \tau(k)$ $v_t \neq \phi_j$. Hence the value of node ϕ_j, which is used to evaluate ϕ_k, must be in memory at step $\tau(i)$. Since i was chosen arbitrarily, we may conclude that the space required by C is not less than $\max_{1 \leq i \leq n} |B^f_\phi(i)|$. The theorem follows by minimizing over all possible $\phi \in \Phi(f)$ and by maximizing over all possible $f \in F_G$. Q.E.D.

3.2 Application to Known CDAGs

In this subsection we illustrate how the marking approach constitutes a uniform and effective framework for proving lower bounds on the space complexity of CDAGs. Specifically, we will show how most of the lower bounds known for relevant CDAGs can be obtained by applying Theorem 1.

Complete binary tree. Let $G = (V, E)$ be a *complete binary tree* with $n = 2m - 1$ nodes, m leaves and arcs directed from each internal node to its parent. Thus, there are m inputs (the leaves) and one output (the root). The following argument is a reformulation of the one given in [PH70], within the marking framework. Let $\phi = \phi_1\phi_2 \cdots \phi_n$ be an arbitrary $f_G^{(sing)}$-marking of G and let $\phi_1\phi_2 \cdots \phi_j$ be the smallest prefix that includes all leaves. Hence ϕ_j must be a leaf. It is easy to see that all nodes in the unique path π from ϕ_j's father to the root must be included in the suffix $\phi_{j+1} \cdots \phi_n$. There are $\log m + 1$ paths from distinct leaves to the root which intersect only at π. Since all leaves are in the prefix and all nodes of π are in the suffix, every such path includes a distinct node in $B_\phi^{f_G^{(sing)}}(j)$. By Theorem 1, we get

$$S_{\text{free}}(G) \geq \log m + 1,$$

which is tight, as shown in [PH70].

Pyramid. Let $G = (V, E)$ be a *pyramid CDAG* with $n = m(m + 1)/2$ nodes obtained by taking the half of an $m \times m$ mesh above and including the nodes on the main diagonal, and by directing all edges towards the upper-right corner of the mesh. The graph has m inputs (the nodes on the mesh diagonal) and one output (the upper-right corner of the mesh). Let $\phi = \phi_1\phi_2 \cdots \phi_n$ be an arbitrary $f_G^{(sing)}$-marking of G. Let also $\phi_1\phi_2 \cdots \phi_j$ be the smallest prefix that includes all inputs. Hence ϕ_j must be an input. By reasoning as for the binary tree (thus reformulating the argument in [Coo74] within the marking framework) we can show that $|B_\phi^{f_G^{(sing)}}(j)| \geq m$. By Theorem 1, this implies

$$S_{\text{free}}(G) \geq m,$$

which is tight [Sav98].

Composed CDAGs. Let $G = (V, E)$ be a CDAG and $G' = (V', E')$ a subdag of G, where $|V| = n$ and $|V'| = m \leq n$. Clearly, any free-input computation of G includes (at least) one free-input computation of G', hence $S_{\text{free}}(G) \geq S_{\text{free}}(G')$.

This fact is reflected in the lower bound stated in Theorem 1, as explained below. It is easy to see that for every marking rule $f_{G'}$ for G' there exists a marking rule f_G for G which is an *extension* of $f_{G'}$ in the sense that $f_G(v) = f_{G'}(v)$ for every node $v \in V'$ which is not an output for G'. Moreover, the $f_{G'}$-markings of G' are in one-to-one correspondence with the projections of the f_G-markings of G on the nodes of V'. Therefore, we have that

$$\max_{f_G \in F_G} \min_{\phi \in \Phi(f_G)} \max_{1 \leq i \leq n} |B_\phi^{f_G}(i)| \geq \max_{f_{G'} \in F_{G'}} \min_{\phi \in \Phi(f_{G'})} \max_{1 \leq i \leq m} |B_\phi^{f_{G'}}(i)|. \quad (1)$$

Let G be the *FFT CDAG* with m inputs, m outputs, $m(\log m + 1)$ nodes and arcs directed from the inputs towards the outputs (a formal definition can be found in [Sav98]). Such a CDAG contains several copies of the m-leaf complete

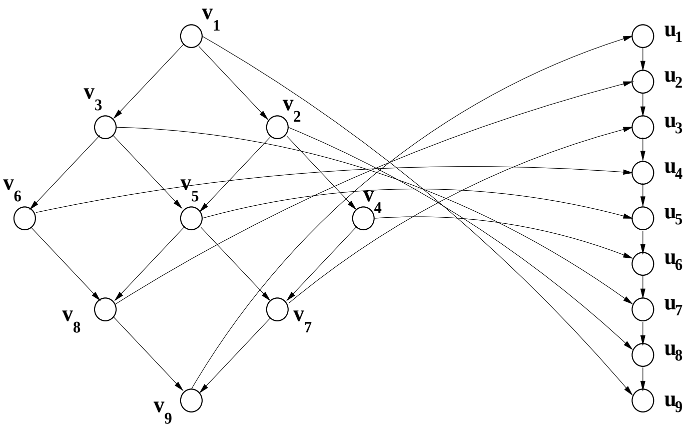

Fig. 1. The diamond/linear array CDAG for $m = 9$.

binary tree G' as subgraph. Relation 1 implies that the marking framework provides a lower bound $S_{\text{free}}(G) \geq \log m + 1$ which, as argued in [Sav98], is tight. As another example consider a *diamond CDAG* G with one input, one output, and $n = m^2$ nodes, obtained by gluing together two $(m(m+1)/2)$-node pyramids at the inputs (corresponding inputs are coalesced into one node) and by reversing the direction of the arcs in the top pyramid. Again, Relation 1 implies that the marking framework provides a lower bound $S_{\text{free}}(G) \geq m$, which is easily seen to be tight.

Finally, we want to show that marking rules can be defined in such a way to bring forward the presence of space demanding subgraphs within a CDAG, and that, in fact, in some cases this is necessary in order to obtain good lower bounds. Let G be the $n = 2m$-node CDAG formed by an m-node diamond CDAG D and an m-node linear array L, connected as follows (see Figure 1). Let $v_1, v_2, \ldots, v_m$ be a topological ordering of the diamond nodes and let $u_1, u_2, \ldots, u_m$ be the (unique) topological ordering of the array nodes. In addition to the arcs of the diamond and those of the linear array, E contains the arcs (v_i, u_{m-i+1}), for $1 \leq i \leq m$. Therefore, G has one input, namely the diamond input v_1, and one output, namely the linear array output u_m. It is easy to see that the space complexity of the free-input computations of G is dominated by that of the diamond component. By selecting $f \in F_G$ as a suitable extension of $f_D^{(sing)}$, one can prove $S_{\text{free}}(G) \geq \sqrt{m}$, which is tight. However, the following claim shows that the singleton marking rule $f_G^{(sing)}$ relative to the entire CDAG G yields only a trivial bound on $S_{\text{free}}(G)$.

Claim. $\min_{\phi \in \Phi(f_G^{(sing)})} \max_{1 \leq i \leq n} |B_\phi^{f_G^{(sing)}}(i)| \leq 2$.

Proof: Let $\phi = \phi_1 \phi_2 \cdots \phi_{2m}$ be such that for $1 \leq i \leq m$

$$\phi_{2i-1} = v_{m-i+1},$$

$$\phi_{2i} = u_i.$$

It is easy to see that ϕ is an $f_G^{(sing)}$-marking. Moreover, $B_\phi^{f_G^{(sing)}}(1) = \{v_m\}$ and, for $1 \leq i < m$,

$$B_\phi^{f_G^{(sing)}}(2i) = \{u_i\},$$
$$B_\phi^{f_G^{(sing)}}(2i+1) = \{v_{m-i}, u_i\}.$$

Thus, the lower bound provided by $f_G^{(sing)}$ is at most 2. Q.E.D.

Unlike the marking rule f mentioned above, which coincides with $f_D^{(sing)}$ on the diamond, $f_G^{(sing)}$ is unable to bring forward the complexity of the diamond due to the "disturbance" introduced by the arcs between D and L.

4 Access Complexity

The notion of access complexity (closely related to the I/O complexity studied in [HK81]) was introduced in [BP99] to capture the cost of CDAG computations in memory hierarchies by counting the number of accesses directed above a certain level. In this section we develop bounds on the access complexity of a CDAG in terms of the space complexity of certain subgraphs. A first method is based on a partition of the given CDAG G chosen a priori and relates the access complexity of G to the space complexity of the subgraphs of such partition. The method is conceptually simple to apply, since only the chosen subgraphs need to be analyzed, and, as an example shows, it may be employed successfully to get tight bounds. Building upon [HK81,Sav93], the remaining two subsections develop a more sophisticated analysis which, while possibly more involved when studying individual CDAGs, does provide almost matching lower and upper bounds on the access complexity of a CDAG G in terms of the minimum cardinality of certain coverings of G into (non necessarily disjoint) subgraphs with limited space complexity.

4.1 The Prepartition Approach

We begin with a simple observation. A standard computation C of a CDAG G can be transformed into a free-input computation C' which starts from an initially empty memory and uses the same schedule of operations as C except that every node evaluation that requires an input as operand is preceded by a `load` of that input. It is easy to see that at least $S_{\text{free}}(G)$ distinct memory cells are accessed during C' and, therefore, during C. Since $\max\{0, S_{\text{free}}(G) - s\}$ of these cells must belong to L2 we have:

$$Q_G(s) \geq \max\{0, S_{\text{free}}(G) - s\}.$$

The following theorem generalizes this argument.

Theorem 2. *Let G be h node-disjoint subgraphs $G_1, G_2, \ldots, G_h$ with space complexities (for free input computations) $S_{\text{free}}(G_i)$, $1 \leq i \leq h$. We have:*

$$Q_G(s) \geq \sum_{i=1}^{h} \max\{0, S_{\text{free}}(G_i) - s\}.$$

Proof: Consider a standard computation C of G and let C_i be the subcomputation of C relative to subgraph G_i, $1 \leq i \leq h$. By reasoning as before we can easily conclude that at least $S_{\text{free}}(G_i)$ distinct memory cells are accessed to read/write values of nodes of G_i. This accounts for at least $\max\{0, S_{\text{free}}(G_i) - s\}$ accesses to L2 relative to nodes of G_i, and the theorem follows. Q.E.D.

As an example, consider a diamond CDAG G with $n = m^2$ nodes. Note that G can be partitioned into $\Theta\left(n/(4s^2)\right)$ node-disjoint diamond subgraphs with $4s^2$ nodes and space complexity (relative to free-input computations) $2s$ each, thus yielding $Q_G(s) = \Omega\left(n/s\right)$. Moreover, one can easily orchestrate a computation of G which indeed performs $O\left(n/s\right)$ access at addresses $x \geq s$.

4.2 The Hong and Kung's Lower Bound Method

Below, we review the methodology proposed by [HK81] for proving lower bounds on the access complexity of CDAGs, based on the notion of s-partition. Let $G = (V, E)$ be an n-node CDAG. A partition of V into subsets $\{V_1, V_2, \ldots, V_h\}$ is called a *topological partition* if, for each $(v, v') \in E$, $v \in V_i$ and $v' \in V_j$ implies $i \leq j$.

Definition 1. *An* s-partition *of G is a topological partition $\{V_1, V_2, \ldots, V_h\}$ of subsets of V such that:*

1. *For every $1 \leq i \leq h$ there is a set $D_i \subseteq V$ of size at most s intersected by every path from an input to a node in V_i. D_i is said to be a* dominator set *for V_i.*
2. *For every $1 \leq i \leq h$ the set $M_i \subset V_i$ of nodes of V_i with no successors in V_i has size at most s. M_i is called the* minimum set *of V_i.*

The following theorem is proved in [HK81, Theorem 3.1].

Theorem 3. *Let $h_G(2s)$ be the minimum cardinality of a $2s$-partition for G. Then, the access complexity of G satisfies*

$$Q_G(s) \geq s(h_G(2s) - 1).$$

We observe that the above result, although successfully employed in [HK81] to prove strong lower bounds for several CDAGs, is rather weak if applied to CDAGs with few inputs and outputs. Specifically, if a CDAG G has at most $2s$ inputs and at most $2s$ outputs, as is the case, for example, of the diamond CDAG, the entire node set V has a dominator set (the set of inputs) and a minimum set (the set of outputs) of size at most $2s$, hence it forms an $2s$-partition of cardinality

$h_G(2s) = 1$. In this case the $s(h_G(2s) - 1)$ lower bound on the access complexity becomes meaningless.*

Savage [Sav98] presents a version of the Hong and Kung result, indicating that his version is weaker but simpler to state and prove. Strictly speaking, Savage's version is not comparable with Hong and Kung's (i.e., each version might lead to stronger bound than the other one on suitable CDAGs). In the following subsection we further develop the ideas of both [HK81] and [Sav98], achieving a result strictly stronger than both, and show that in an interesting sense the resulting bound is tight.

4.3 The Covering Method: Lower and Upper Bounds

Let $G = (V, E)$ be an n-node CDAG. We observe that, when partitioning V into subsets, the crucial measure for a subset V_i is the amount of space needed to compute the nodes in V_i starting from a suitably chosen subset D_i of their ancestors (which is easily seen to be a dominator set of V_i). Generally, in the process, a larger set U_i of nodes is evaluated.

Definition 2. *A topological partition $\{V_1, V_2, \ldots, V_k\}$ of V is s-*coverable *if for $1 \leq i \leq k$, there exists a set U_i, with $V_i \subseteq U_i \subseteq \cup_{j=1}^{i} V_j$, such that $S(U_i) \leq s$, considering as inputs of U_i all nodes of in-degree 0 or with a predecessor outside U_i. (Note that the inputs of U_1 are a subset of the inputs of G.)*

Theorem 4. *Let $k_G(2s)$ be the minimum cardinality of a $2s$-coverable partition for G. Then the access complexity of G satisfies*

$$Q_G(s) \geq s(k_G(2s) - 1)$$

Proof: Consider a T-step computation of G. For every $v \in V$ we mark the step of its first evaluation (inputs are all marked at the beginning of the computation). Clearly, each node of V is associated with *exactly* one marked step. Partition the interval $[1, T]$ relative to the computation of G into K subintervals such that in the first $K - 1$ subintervals exactly s accesses to L2 are performed, and in the K-th subinterval at most s accesses to L2 are performed. For $1 \leq i \leq K$ we let V_i consist of the nodes associated with the steps marked in the i-th subinterval I_i. It is easy to see that $\{V_1, V_2, \ldots, V_K\}$ is a topological partition. To show that it is a $2s$-coverable, we let U_i be the union of two (possibly nondisjoint) sets of nodes: the set of nodes whose values, at the beginning of I_i, reside in L1 or in the (at most) s cells of L2 accessed during I_i; and the set of nodes evaluated during I_i. It is easy to see that the inputs of U_i are included in the former set and that $V_i \subseteq U_i \subseteq \cup_{j=1}^{i} V_j$. Moreover, it is easy to argue that there exists a standard computation of U_i that uses space at most $2s$. The theorem follows since the evaluation of G performs at least $s(K - 1)$ accesses to L2 and K is bounded from below by $k_G(2s)$. Q.E.D.

* In fact, an $\Omega(n/s)$ lower bound on the access complexity of a variant of the n-node diamond is proved in [HK81], assuming that all $2\sqrt{n} - 1$ nodes on the upper sides of the diamond are inputs.

The following theorem provides an upper bound on the access complexity of general CDAGs.

Theorem 5. *Let G have a $2s$-coverable partition of cardinality k. Then,*

$$Q_G(2s) \leq 4sk.$$

Proof: Let $\{V_1, V_2, \ldots, V_k\}$ be a $2s$-coverable partition of G. Let L1 be the set of memory locations of address smaller than $2s$ and L2 the set of the remaining locations. We can compute G in k *stages*, where in Stage i the following activities are performed, $1 \leq i \leq k$:

1. Download the inputs of U_i from L2 to L1 (except for Stage 1);
2. Compute U_i in L1, uploading to L2 the value of every node v belonging to some U_j, with $j > i$, that was not uploaded in previous stages.

The correctness of the computation follows immediately from the definition of $2s$-coverable partition. Moreover, it is clear that each input of U_i contributes at most two accesses to L2 (one associated with its upload and one at the beginning of Stage i), while nodes that are not inputs of some U_i never pass through L2. Let $D_i \subseteq U_i$ be the set of inputs of U_i and note that by definition of $2s$-partition $|D_i| \leq 2s$. We have that the access complexity of the computation is at most

$$2\left(\sum_{i=1}^{k} |D_i|\right) \leq 4sk.$$

Q.E.D.

As a straighforward corollary we have that $Q_G(2s) \leq 4sk_G(2s)$. Moreover, from Theorem 4 we have that $Q_G(s) \geq s(k_G(2s) - 1)$. As observed in several studies (e.g., [AACS87]), relations on Q_G of this type translate into tight upper and lower bounds on running times for memory hierarchies where access time varies smoothly with the address.

5 Conclusions

In this paper, we have studied some issues related to space and access complexity. We have developed a lower bound technique for space of wide applicability. One interesting open issue remains whether our technique could reproduce the $\Omega(n/\log n)$ lower bound obtained in [PTC77] for a particular CDAG recursively constructed out of superconcentrators. Attempts to resolve this issue might shed further light on the relation of space complexity to other graph-theoretic properties of CDAGs. We have also explored various relations between access complexity and space complexity. Here the attention was focused on minimizing $Q_G(s)$ for a fixed value of s. When studying computations in hierarchical memory models, such as the HMM of [AACS87] or the HRAM of [BP99], the objective is the minimization of running time, which can be viewed as a functional of the function $Q_G(\cdot)$. A number of interesting questions arise in this connection, which can benefit of our results on the access complexity, but also from a deeper investigation of the possible shapes for the function Q (e.g., see [BP00] for issues related to the portability of performance across different hierarchical machines).

References

[AACS87] A. Aggarwal, B. Alpern, A.K. Chandra, and M. Snir. A model for hierarchical memory. In *Proc. of the 19th ACM Symp. on Theory of Computing*, pages 305–314, 1987.

[AV88] A. Aggarwal and J.S. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1127, 1988.

[BP99] G. Bilardi and F.P. Preparata. Processor-time tradeoffs under bounded-speed message propagation: Part II, lower bounds. *Theory of Computing Systems*, Vol. 32, 531-559, 1999.

[BP00] G. Bilardi and E. Peserico. Efficient portability across memory hierarchies. *Manuscript*, 2000.

[Coo74] S.A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9:308–316, 1974.

[HK81] J.W. Hong and H.T. Kung. I/O complexity: The red-blue pebble game. In *Proc. of the 13th ACM Symp. on Theory of Computing*, pages 326–333, 1981.

[HPV77] J. Hopcroft, W. Paul, and L. Valiant. On time versus space. *Journal of the ACM*, 24(2):332–337, April 1977.

[LT82] T. Lengauer and R.E. Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *Journal of the ACM*, 29(4):1087–1130, October 1982.

[PH70] M.S. Paterson and C.E. Hewitt. Comparative schematology. In *Proc. of Project MAC Conf. Concurrent Systems and Parallel Computation*, pages 119–127, Woods Hole, MA, 1970.

[Pip80] N. Pippenger. Pebbling. Technical report, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, 1980. See also *5th IBM Symposium of Mathematical Foundations of Computer Science.*

[PTC77] W.J. Paul, R.E. Tarjan, and J.R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[RS83] A.L. Rosenberg and I.H. Sudborough. Bandwidth and pebbling. *Computing*, 31:115–139, 1983.

[Sav93] J.E. Savage. Space-time tradeoffs in memory hierarchies. Technical Report CS-93-08, Department of Computer Science, Brown University, 115 Waterman St. Providence, RI 02912-1910, 1993.

[Sav98] J.E. Savage. *Models of Computation – Exploring the Power of Computing.* Addison Wesley, Reading, MA, USA, 1998.