Brief Announcement: Approximating the I/O Complexity of One-Shot Red-Blue Pebbling

Timothy Carpenter*
Dept. of Computer Science
and Engineering
The Ohio State University
Columbus, OH USA
carpenter.454@osu.edu

P. Sadayappan†
Dept. of Computer Science
and Engineering
The Ohio State University
Columbus, OH USA
saday@cse.ohio-
state.edu

Fabrice Rastello
Inria
Univ. Grenoble Alpes, LIG,
F-38000 Grenoble, France
CNRS, LIG, F-38000
Grenoble, France
Fabrice.Rastello@inria.fr

Anastasios Sidiropoulos*
Dept. of Computer Science
and Engineering
Dept. of Mathematics
The Ohio State University
Columbus, OH USA
sidiropoulos.1@osu.edu

ABSTRACT

Red-blue pebbling is a model of computation that captures the complexity of I/O operations in systems with external memory access. We focus on one-shot pebbling strategies, that is without re-computation. Prior work on this model has focused on finding upper and lower bounds on the I/O complexity of certain families of graphs. We give a polynomial-time bi-criteria approximation algorithm for this problem for graphs with bounded out-degree. More precisely, given a n -vertex DAG that admits a pebbling strategy with R red pebbles and I/O complexity opt , our algorithm outputs a strategy using $O(R \cdot \log^{3/2} n)$ red pebbles, and I/O complexity $O(\text{opt} \cdot \log^{3/2} n)$. We further extend our result to the generalization of red-blue pebble games that correspond to multi-level memory hierarchies. Finally, we complement our theoretical analysis with an experimental evaluation of our algorithm for red-blue pebbling.

Keywords

red-blue pebble game; I/O complexity; memory hierarchy; approximation algorithms

1. INTRODUCTION

Pebbling games are a powerful abstraction for analyzing

*Supported by NSF under award CAREER-1453472 and grant CCF-1423230.

†Supported in part by DOE award DE-SC0012489

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SPAA '16 July 11-13, 2016, Pacific Grove, CA, USA

© 2016 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4210-0/16/07.

DOI: <http://dx.doi.org/10.1145/2935764.2935807>

the complexity of computations [13, 12]. We focus on the *red-blue* pebble game, as introduced in [7], which is used to capture the I/O complexity of computations in the presence of external memory. We are interested in *one-shot* pebbling, that is without re-computation.

The rules of the red-blue pebble game are as follows: The input consists of a DAG G . Initially, all input vertices (i.e. with in-degree zero) have a blue pebble. (R1) A red pebble can be placed in any vertex that contains a blue pebble. (R2) A blue pebble can be placed in any vertex that has a red pebble. (R3) A red pebble can be placed in any vertex whose all immediate predecessors have red pebbles. (R4) A pebble can be removed from any vertex. A terminal configuration is one where all output vertices (i.e. of out-degree zero) have blue pebbles. Intuitively, red pebbles correspond to (limited) internal memory, while blue pebbles correspond to (unlimited) external storage. The *I/O complexity* of a pebbling is the number of times that transitions according to rules (R1) and (R2) are performed. One-shot pebbling means that rule (R3) can be performed at most once for each vertex. We remark that this is also known as *red-blue-white* pebbling (see [5] for details).

Most prior work on red-blue pebble games is focused on obtaining upper and lower bounds on the I/O complexity of specific families of graphs, such as pyramids [14], diamonds, butterflies, matrix multiplication graphs, and so on [5].

Recent results in pebbling study the approximability of the minimum number of pebbles required for a given input DAG, for various pebble games [2, 12]. We obtain a bi-criteria approximation algorithm for the I/O complexity of red-blue pebble games with a given number of red pebbles. More precisely, let $\text{opt}_R(G)$ be the minimum I/O complexity for red-blue pebbles of G with R red pebbles. We give a polynomial-time algorithm which given a DAG G of bounded out-degree and some $R > 0$, outputs a pebbling strategy for G using $O(R \cdot \log^{3/2} n)$ red pebbles, with I/O complexity $O(\text{opt}_R(G) \cdot \log^{3/2} n)$.

We further extend our algorithm to the generalization

of red-blue pebble games for multi-level memory hierarchy given in [15].

Finally, we evaluate our algorithm experimentally. In some cases we obtain pebbling strategies that outperform the best-known theoretical upper bounds.

2. PRELIMINARIES

A *directed vertex separator* of a directed graph $G = (V, E)$ is a partition (A, B, S) of V such that there are no edges between A and B , and all edges between S and B are directed from S to B . We define the *size* of the separator to be $|S|$. We say that (A, B, S) is α -balanced, for some $0 < \alpha < 1$ if $\max\{|A|, |B|\} \leq \alpha|V|$.

It can be shown that such separators exist in any DAG G by splitting some topological ordering into sets A, B around any point in the ordering, and moving any vertex of A with a child in B to the separator S . Using an approximation algorithm for directed sparsest-cut and the reduction described in [10], it is possible to find a directed balanced separator (A, B, S) of G which approximately minimizes the ratio $|S| / \min\{|S \cup A|, |S \cup B|\}$.

A bi-criteria ρ -approximation for minimum directed balanced vertex separator is an algorithm that outputs a directed α -balanced separator, of size at most $\rho \cdot b$, where b is the size of a minimum $1/2$ -balanced directed vertex separator, for some universal constant $\alpha > 0$. For the remainder of this paper, let $\alpha = 1/4$, and let $\rho^* > 0$ be such that there exists a polynomial time bi-criteria ρ^* -approximation algorithm for minimum directed balanced vertex separator. The actual value of ρ^* will depend on the class of graphs under consideration. For some $X \subseteq V$ we denote by $G[X]$ the vertex-induced subgraph of G on X .

3. THE APPROXIMATION ALGORITHM

The following is a modified version of the algorithm in [10] for approximating the minimum black pebbling cost.

Algorithm 1 Ordering algorithm

Input: DAG G .

Output: Ordering of V .

If $|V| \leq 1$ then output the unique ordering of V .

Otherwise, compute a bi-criteria ρ -approximate directed $1/4$ -balanced vertex separator (A, B, S) of G . Recursively order $G[A \cup S]$ and $G[B]$, and output the ordering of G as the ordering of $G[A \cup S]$ followed by $G[B]$.

We now describe our algorithm for one-shot red-blue pebbling: We first use Algorithm 1 to find an ordering π of V . The order in which vertices are first pebbled red is determined by π . If at some step of the pebbling there are no unused red pebbles when one is needed, then from the set of vertices with red pebbles, we find the vertex v with the longest span before the next child will be pebbled (ties broken arbitrarily), and we pebble v blue. For all $v, v' \in V$ such that $\pi(v') = \pi(v) + 1$, after pebbling v and before pebbling v' , pebble red any parents of v' that are pebbled blue. When all the children of a vertex v have been pebbled, the red pebble is removed from v . This completed the description of the red-blue pebbling algorithm.

THEOREM 1. *Let $G = (V, E)$ be a n -vertex DAG with maximum out-degree d . Then the red-blue pebbling algorithm on G outputs a pebbling strategy using $O(R \cdot \rho^* \cdot \log n)$ red pebbles with I/O complexity $O(d \cdot \rho^* \cdot \text{opt}_R(G) \cdot \log n)$.*

From [1] we have $\rho^* = O(\sqrt{\log n})$. Thus we obtain the following.

COROLLARY 1. *Let G be a n -vertex directed acyclic graph with maximum out-degree d . There exists a polynomial-time algorithm that for any $R > 0$, computes a red-blue pebbling strategy for G using $O(R \cdot \log^{3/2} n)$ red pebbles with I/O complexity $O(\text{opt}_R(G) \cdot d \cdot \log^{3/2} n)$.*

4. EXTENSION TO MULTI-LEVEL MEMORY HIERARCHIES

The memory hierarchy model extends red-blue pebbling by adding several types of pebbles, organized into levels $1, \dots, k$. Let $P_1, \dots, P_k$ be the number of pebbles at each level, and $P_k = \infty$. Each level $i > 1$ has an I/O cost c_i associated with using a pebble. The following rules govern the placement and removal of pebbles from vertices: (M1) A pebble on level $1 < i < k$ can only be placed on a vertex to replace a pebble of level $i - 1$ or $i + 1$. A level-1 pebble can only be placed (M2) on a source vertex, (M3) a vertex with a level-2 pebble, or (M4) a vertex whose parents all hold a level-1 pebble. (M5) A level- k pebble may only be placed on a vertex holding a level- $(k - 1)$ pebble. Pebbles of any level may be removed at any time. The pebbling is completed only when all vertices have been pebbled. Notice that when $k = 2$, the memory hierarchy model coincides with red-blue pebbling. The one-shot version of this model is obtained by allowing rule (M4) to be performed at most once per vertex.

We now describe our algorithm for finding a hierarchical pebbling strategy. Let $G = (V, E)$ be the input DAG. We first use Algorithm 1 to find a topological ordering π of G . The ordering π determines the order in which each vertex is pebbled. If v is the next vertex to be pebbled, we ensure that all parents of v hold a level-1 pebble. If there is only one level-1 pebble available when one is needed, we free up an addition pebble as follows: From the set of vertices holding a level-1 pebble, identify a vertex u with the longest span in π before the next child of u is pebbled, and transition u to level 2. When any $u \in V$ is being transitioned to a level- i pebble, we consider the following cases:

- (1) There are at least two level- i pebbles available. In this case, give u a level- i pebble.
- (2) There is only one level- i pebble available.
 - (2.1) There is a vertex r with a level- i pebble, and r has a longer span than u . Transition r to a level- $(i + 1)$ pebble, and give u a level- i pebble.
 - (2.2) There is no vertex r with a level- i pebble and longer span than u . Pebble u using the last level- i pebble, and transition u to a level- $(i + 1)$ pebble.

For all $v \in V$, once the last child of v has been pebbled, release the pebble v holds. This completes the description of the algorithm.

Let $\text{opt}(G)$ be the minimum total I/O complexity of any pebbling strategy for G with $P_1, \dots, P_{k-1}$ pebbles at level $1, \dots, k - 1$. The following Theorem summarizes our result on the performance of the above algorithm.

Figure 1: Performance of Algorithm 1 on the diamond graph of size 50×50 (left) and the 1-dimensional Jacobi graph on 2500 vertices (right).

THEOREM 2. Let G be a n -vertex DAG with maximum out-degree d . The hierarchical pebbling algorithm for G outputs a pebbling strategy using $O(P_i \cdot \rho^* \cdot \log n)$ level- i pebbles for each $i < k$ and has I/O complexity $O(d \cdot \rho^* \cdot \text{opt}(G) \cdot \log n)$.

5. EXPERIMENTAL EVALUATION

We have experimentally evaluated our algorithm on various DAGs. The most computationally intensive step is finding balanced directed vertex separators. Unfortunately, most widely available graph partitioning software works only on undirected graphs¹. We have implemented three algorithms for this step: The first one is the directed version of the approximation algorithm due to [11], using the faster combinatorial algorithm for maximum concurrent flow from [16]. The second one is a heuristic based on the METIS graph partitioning library for undirected graphs [8]. The third implementation is a heuristic based on spectral partitioning of undirected graphs.

Figure 1 depicts the results on the 50×50 diamond graph (i.e. the DAG obtained from the grid by orienting all edges from left to right and from top to bottom) and the 1-D Jacobi graph. We observe that, in either case, the choice of graph partitioning algorithm does not significantly affect the I/O complexity. Figure 1 also includes the best previously known theoretical upper and lower bounds (see [5]) for the diamond graph. Interestingly, our implementations using METIS and spectral partitioning outperform the theoretical upper bound when the number of red pebbles is at least 45.

6. REFERENCES

- [1] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. $O(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proc. of the 37th Annual ACM Symposium on Theory of Computing*, STOC '05, 2005.
- [2] P. Austrin, T. Pitassi, and Y. Wu. Inapproximability of treewidth, one-shot pebbling, and related layout problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 13–24. Springer, 2012.
- [3] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner. Graph partitioning and graph clustering - 10th DIMACS implementation challenge workshop.
- [4] S. M. Chan, M. Lauria, J. Nordstrom, and M. Vinyals. Hardness of approximation in PSPACE and separation results for pebble games. In *Proc. of IEEE 56th Ann. Symp. on Found. of Comp. Science*, 2015.
- [5] V. Elango, F. Rastello, L.-N. Pouchet, J. Ramanujam, and P. Sadayappan. Data access complexity: The red/blue pebble game revisited. Technical report, OSU-CISRC-7/13-TR16, Ohio State University, 2013.
- [6] U. Feige, M. Hajiaghayi, and J. R. Lee. Improved approximation algorithms for minimum-weight vertex separators. In *Proc. of the 37th Annual ACM Symp. on Theory of Comp.*, STOC '05, 2005.
- [7] J.-W. Hong and H. T. Kung. I/O complexity: The red-blue pebble game. In *Proc. of the 13th Annual ACM Symposium on Theory of Computing*, 1981.
- [8] G. Karypis and V. Kumar. Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0. 1995.
- [9] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, December 1998.
- [10] P. Klein, A. Agrawal, R. Ravi, and S. Rao. Approximation through multicommodity flow. In *Proc. of the 30th Ann. Symp. on Found. of Comp. Science*, 1989.
- [11] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.
- [12] J. Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. *Foundations and Trends in Theoretical Computer Science*. <http://people.csail.mit.edu/jakobn>, 2010.
- [13] N. Pippenger. *Advances in pebbling*. Springer, 1982.
- [14] D. Ranjan, J. Savage, and M. Zubair. Upper and lower I/O bounds for pebbling r-pyramids. *Journal of Discrete Algorithms*, 14:2–12, 2012.
- [15] J. E. Savage. Extending the hong-kung model to memory hierarchies. In *Computing and Combinatorics*, pages 270–281. Springer, 1995.
- [16] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. *Journal of the ACM (JACM)*, 37(2):318–334, 1990.

¹This is one of the main open problems emphasized in the 10th DIMACS Implementation Challenge [3].