Communication Efficient Data Structures on the BSP model with Applications in Computational Geometry *

Alexandros V. Gerbessiotis and Constantinos J. Siniolakis

Computing Laboratory, Oxford University, Wolfson Building, Oxford OX1 3QD, UK.

Abstract. The implementation of data structures on distributed memory models, like the Bulk-Synchronous Parallel (BSP), rather than shared memory ones, like the PRAM, offers a serious challenge. In this paper we undertake the architecture independent study of the communication and synchronization requirements of searching "ordered h-level graphs", which include most of the standard data structures. We propose n-way search as a general tool for the design, analysis, and implementation of BSP algorithms. This technique allows elegant high-level design and analysis of algorithms, using data structures similar to that of sequential models. Our methods are within a $1 + o(1)$ factor of the respective sequential methods. An application to computational geometry is also presented.

1 Introduction and the BSP Model

The implementation of data structures on distributed memory models such as the Bulk-Synchronous Parallel rather than shared memory ones offers a serious challenge. The problem of realizing such distributed data structures has been addressed in the past and several results have been obtained for particular network architectures, i.e., hypercubes, meshes. Architecture dependent features involved in these implementations, make the migration of these results to the more general setting of the BSP model non-trivial. The only previously known result on searching [1] involves a restricted subset of the graphs we present in this paper. In [1], searching on static balanced d-ary trees is realized optimally on an augmented BSP model. In this work we generalize and extend this result in several ways, as described in later sections. Particularly, the main technical contribution of this paper is the architecture independent study of the communication and synchronization requirements of a major problem [2] namely, that of realizing n search processes on "ordered h-level graphs", which include most of the standard data structures. Along the lines of [2], we propose n-way search as a general tool for the design, analysis, and implementation of BSP algorithms. This technique allows elegant high-level design and analysis of algorithms, using

* The work of the first author was supported in part by EPSRC(UK) under grant GR/K16999, and the second author was supported in part by a Bodossaki Foundation Graduate Scholarship.

data structures similar to that of sequential models, and, provided that the underlying data structure is an ordered h-level graph, our methods are within an $1 + o(1)$ factor of the respective sequential methods. In this version of the paper proofs are omitted, we refer to [4] for a detailed discussion.

The computational model adopted is the Bulk-Synchronous Parallel (BSP) model of computation, which deals explicitly with the notion of communication and synchronization among computational threads. The BSP model was originally proposed by L.G. Valiant [6] and subsequently addressed in [3, 5] to bridge the software and hardware worlds with respect to parallel computing. For a detailed discussion of the BSP model, we refer to [6].

We specify the performance of BSP algorithms in terms of parameters π and μ introduced in [3]. Briefly π (respectively, μ) is the ratio of total parallel computation (respectively, communication) work over sequential work. Throughout the paper, lg and log denote logarithms to the base 2 and e respectively.

2 Ordered h-level Graphs

We study the communication and synchronization requirements of searching on "ordered h-level graphs" [2], which include most of the standard data structures. We propose multi-way searching as a general tool for the design, analysis, and implementation of BSP algorithms. We briefly define ordered h-level graphs, refer to [4] for a detailed discussion. The definitions and notation of this section are derived from [2]. We extend, however, some of the notions presented in [1, 2] to include a more general class of ordered h-level graphs.

Let $G = (V, E)$ be a directed graph with vertex set V and edge set E. An *ordered h-level partitioning* of G is a partitioning of V into h ordered sequences of disjoint sets $V_1, V_2, \ldots, V_h$, together with an ordering of the elements in each subset V_i, $i \in \{1, 2, \ldots, h\}$. For every h-partitioning of G we define, $\forall\, v \in V$, two parameters, namely $Level(v)$ and $Level\_Index(v)$ as follows: (a) $Level(v) = l$ if and only if $v \in V_l$, $l \in \{1, 2, \ldots, h\}$, (b) $Level\_Index(v)$ is the rank of v with respect to the ordering of the vertices in $V_{Level(v)}$. A directed graph $G = (V, E)$ is called an *ordered h-level graph* if it has the following properties: (a) every vertex v in V has out degree of at most d, where d is some positive integer (in contrast to [2] we do not insist on constant degree graphs), (b) the number of sources s of G is at most d, (c) there exists an ordered h-partitioning $V_1, V_2, \ldots, V_h$, of G such that (i) every source of G is contained in V_1, (ii) if (u, v) is an edge of G, then $Level(v) = Level(u) + 1$, (iii) if (u, v) and $(\bar{u}, \bar{v})$ are two edges of G with $Level(u) = Level(\bar{u})$ and $Level\_Index(u) < Level\_Index(\bar{u})$, then $Level(v) = Level(\bar{v})$ and $Level\_Index(v) \leq Level\_Index(\bar{v})$, (d) the graph has no double edges. As it is noticed in [2], any d-ary tree can be expressed as an ordered h-level graph. The following lemma captures two important facts about ordered h-level graphs.

Lemma 1. *Let $G = (V, E)$ be an ordered h-level graph. The following statements are true: (a) G is planar and acyclic, (b) $|E| \leq |V_1| + 2|V_2 \cup \ldots \cup V_{h-1}| + |V_h| - (h-1)$.*

2.1 n-Way Search on Ordered h-level Graphs

In this section we sketch a p-processor BSP scheme for efficiently searching an ordered h-level graph. Let $G = (V, E)$ be an ordered h-level graph of maximum out-degree d, with vertex set $V = V_1 \cup V_2 \cup \ldots \cup V_h$ and edge set E, where $|V| = m$. Moreover, let U be a universe of possible search queries on G. A *search path* for a query $q \in U$ is a sequence $Path(q) =< v_1, v_2, \ldots, v_h >$ of h vertices of G, $v_i \in V_i$, prescribed by a *next* function $f : U \times (V \cup \{start\}) \rightarrow \{1, 2, \ldots, d\}$ as follows: (a) $f(q, start) = Index(v_1)$ (b) $f(q, v_i) = Index(v_{i+1})$, $i \in \{1, 2, \ldots, h-1\}$, where $(v_i, v_{i+1}) \in E$ and for every $v \in V$, $Index(v) = \sum_{i=1}^{Level(v)-1} |V_i| + Level\_Index(v)$. We do not assume that the search path is given in advance, only that it is constructed during the search process by successive applications of function f. We note that, in contrast to [1], our method has the advantage of constructing the entire search path for each query q and therefore can be applied to the solution of a class of geometric problems such as those presented in [2]. The *n-way search problem* is the following one. Given a set of queries $Q = \{q_1, q_2, \ldots, q_n\} \subseteq U$ and an ordered h-level graph G with $|V| = m$, determine, all n search paths induced by the n queries. In order to achieve communication efficiency on a BSP machine, we maintain a novel distribution of the graph G among the processors. The time complexity of an arbitrary phase l is summarized in the following proposition.

Proposition 2. *Let $Q = \{q_1, q_2, \ldots, q_n\}$ be a set of n queries and let $G = (V, E)$ be an ordered h-level graph, with vertex set $V = V_1 \cup V_2 \cup \ldots \cup V_h$ and edge set E, where $|V| = m \leq n$. Let the maximum out-degree of such a graph be $d = o(n/p)$ and the number of source vertices be $s \leq d$. If f is the next function, and $\tau_l(f)$ is the time required to evaluate f at level l, then, there exists a BSP algorithm for computing the next vertex in the search path of q_i, $\forall\, i \in \{1, 2, \ldots, n\}$, that for any level l, and using the charging policy of section 1, has*

$$\pi = \frac{p}{\tau_l(f)n}\left(\tau_l(f)\left(\left\lceil\frac{n}{p}\right\rceil + \left\lceil\left\lceil\frac{2m}{p}\right\rceil/\delta\right\rceil + o\left(\frac{n}{p}\right)\right) + O\left(\frac{n}{p} + \frac{m}{p\delta} + L\lg p\right)\right),$$

$$\mu = g\frac{p}{\tau_l(f)n}O\left(\delta\frac{n}{p} + \frac{m}{p\delta} + \frac{L}{g}\lg p\right),$$

where δ is some positive integer, that expresses a trade-off between computation and communication time. The algorithm requires $O(\delta n/p + m/p)$ space.

In the description of the previous proposition we used the parameter δ to express a trade-off between computation and communication time, where one-optimality in computation can be achieved for $\delta = \omega(1)$. By using a load-balancing scheme, we achieve one-optimality in computation and yet require $\delta < 3$, as stated in the following theorem.

Theorem 3. *Let $Q = \{q_1, q_2, \ldots, q_n\}$ be a set of n queries and let $G = (V, E)$ be an ordered h-level graph, with vertex set $V = V_1 \cup V_2 \cup \ldots \cup V_h$, edge set E, maximum out-degree $d = o(n/p)$ and $s \leq d$ source vertices such that $|V| \leq n$.*

If f is the next function and $\tau(f)$ is the time required to evaluate f, then, there exists a BSP algorithm for computing all n search processes induced by the n queries, that for all positive integers n, p, and L satisfying $n = \omega(p \lg p)$, and $L = o(\tau(f)n/(p \lg p))$ has $\pi = 1 + O(1/\tau(f))$, and if $g = o(\tau(f))$ also, $\mu = o(1)$.

For $m \geq n$, the deterministic scheme implied in section 2.1 proves to be inefficient, since its running time depends on m. We remove all dependencies on m in the time bound of theorem 3 by introducing randomization. In the full version of the paper [4] we describe the details of the randomized scheme.

2.2 n-Way Point-Location through n-Way Search

We shall now apply the methods we developed in the previous sections to obtain an efficient BSP solution to an important computational geometry, problem, that is, given a set of n points (queries) $Q = \{q_1, q_2, \ldots, q_n\}$ and an ordered partitioning Y of a two-dimensional horizontal strip V by $m-1$ non-intersecting line segments $S = < s_1, s_2, \ldots, s_{m-1} >$ into m slabs, determine, for each $q \in Q$, the vertical slab it belongs to. The solution to this problem, which we call *n-way point-location*, is central to several other important geometric problems. The following theorem summarizes our results on n-way point-location.

Theorem 4. *Let $Q = \{q_1, q_2, \ldots, q_n\}$ be a set of n points (queries) and let Y be an ordered partitioning of a two-dimensional horizontal strip V, such that $|Y| \leq n$. If ϵ is a positive constant such that $0 < \epsilon < 1$, then, there exists a BSP procedure for solving the n-way point-location problem induced by the n queries, that for all positive integers n, p, L, and d satisfying $n = \omega(p \lg p)$, $L = o(n \lg d/(p \lg p))$, and $d = (n/p)^\epsilon$ has $\pi = 1 + o(1)$, and if $g = o(\lg d)$ also, $\mu = o(1)$.*

References

1. A. Baumker, W. Dittrich, and F. Meyer auf der Heide. Truly Efficient Parallel Algorithms: c-Optimal Multisearch for an Extension of the BSP Model. *Annual European Symposium on Algorithms*, 1995.
2. F. Dehne and A. Rau-Chaplin. Implementing data structures on a hypercube multiprocessor, and applications in parallel computational geometry. *Journal of Parallel and Distributed Computing,* 8:367-375, 1990.
3. A. V. Gerbessiotis and L. G. Valiant. Direct bulk synchronous parallel algorithms. *Journal of Parallel and Distributed Computing,* 22:251-267, 1994.
4. A. V. Gerbessiotis and C. J. Siniolakis. Communication Efficient Data Structures on the BSP model with Applications. Technical Report, Computing Laboratory, Oxford University, May 1996.
5. W. F. McColl. General purpose parallel computing. In *Lectures on parallel computation,* (Gibbons, A. and Spyrakis, P., eds.), Cambridge University Press, 1993.
6. L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM,* 33(8):103-111, August 1990.

This article was processed using the LaTeX macro package with LLNCS style