Truly Efficient Parallel Algorithms: c-Optimal Multisearch for an Extension of the BSP Model*
(Extended Abstract)

Armin Bäumker, Wolfgang Dittrich
and Friedhelm Meyer auf der Heide

Department of Mathematics and Computer Science
and Heinz Nixdorf Institute, University of Paderborn
33095 Paderborn, Germany

Abstract
In this paper we design and analyse parallel algorithms with the goal to get exact bounds on their speed-ups on real machines. For this purpose we define an extension of Valiant's BSP model, BSP*, that rewards blockwise communication, and uses Valiant's notion of c-optimality. Intuitively a c-optimal parallel algorithm for p processors achieves speed-up close to \( \frac{p}{c} \). We consider the Multisearch problem: Assume a strip in 2D to be partitioned into m segments. Given n query points in the strip, the task is to locate, for each query, its segment. For \( m \leq n \) we present a deterministic BSP* algorithm that is 1-optimal, if \( n = \Omega(p \log^2 p) \). For \( m > n \), we present a randomized BSP* algorithm that is \( (1 + \delta)\)-optimal for arbitrary \( \delta > 0 \), \( m \leq 2^p \) and \( n = \Omega(p \log^2 p) \). Both results hold for a wide range of BSP* parameters where the range becomes larger with growing input sizes \( m \) and \( n \). We further report on implementation work in progress. Previous parallel algorithms for Multisearch were far away from being c-optimal in our model and do not consider blockwise communication.

1 Introduction
The theory of efficient parallel algorithms is very successful in developing new original algorithmic ideas and analytic techniques to design and analyse efficient parallel algorithms. For this purpose the PRAM has proven to be a very convenient computation model, because it abstracts from communication problems. On the other hand, the asymptotic results achieved only give limited information about the behaviour of the algorithms on real parallel machines. This is (mainly) due to the following reasons.

- The PRAM cost model (communication is as expensive as computation) is far away from reality, because communication is by far more expensive than internal computation on real parallel machines [6].
- The number of processors \( p \) is treated as an unlimited resource, (like time and space in sequential computation) whereas in real machines \( p \) is small (a parallel machine (MIMD) with 1000 processors is already a large machine).

There are several approaches to define more realistic computation models and cost measures to overcome the first objection mentioned above: The BSP model due to

* email: {abk,dittrich,fmadh}@uni-paderborn.de, Fax: +49-5251-603514. Supported in part by DFG-Sonderforschungsbereich 1811 "Massive Parallelität: Algorithmen, Entwurfsmethoden, Anwendungen", by DFG Leibniz Grant Me872/6-1, and by the Esprit Basic Research Action Nr 7141 (ALCOM II)
Valiant [14], the LogP model due to Culler et al. [6], the BPRAM of Aggarval et al. [1], and the CGM due to Dehne et al. [7] to name a few. Note that most of the models except the BPRAM neglect the negative effects of communicating small packets.

To deal with the second objection, Kruskal et al. [10] have proposed a complexity theory which considers speed-up. Valiant has proposed a very strong notion of work optimality of algorithms, \textit{c-optimality}. It gives precise information about the possible speed-up on real machines, the speed-up of a \textit{c}-optimal algorithm should be close to \( p/c \). Besides [8] and [5] there are seemingly no systematic efforts undertaken to design parallel algorithms with respect to this strong optimality criterion.

The computation model used in this paper is the BSP enhanced by a feature that rewards blockwise communication. We design and analyze two algorithms for a basic problem in computational geometry, the Multisearch problem. Our first algorithm is deterministic and 1-optimal. It works for the case of many search queries compared to the number of segments. The second algorithm is designed for the case if only few search queries are asked. It is randomized and proven to be \((1 + \delta)\)-optimal with high probability, for \( \delta > 0 \) arbitrary. Both results hold for wide ranges of BSP* parameters.

1.1 The Multisearch Problem

Multisearch is an important basic problem in computational geometry. It is the core of e.g. planar point location algorithms, segment trees and many other data structures.

Given an ordered \textit{universe} \( U \) and a partition of \( U \) in \textit{segments} \( S = \{s_1, \ldots, s_m\} \). The segments are ordered in the sense that, for each \( q \in U \) and segment \( s_i \), it can be determined with unit cost whether \( q \in s_i \), \( q \in \{s_1 \cup \ldots \cup s_{i-1}\} \), or \( q \in \{s_{i+1} \cup \ldots \cup s_m\} \).

We assume that, initially, the segments and queries are evenly distributed among the processors. Each processor has a block of at most \( \lfloor m/p \rfloor \) consecutive segments and arbitrary \( \lfloor n/p \rfloor \) queries, as part of the input. The \textit{Multisearch problem} is: Given a set of queries \( Q = \{q_1, \ldots, q_n\} \subseteq U \) and a set of segments \( S = \{s_1, \ldots, s_m\} \), find, for each \( q_i \), the segment it belongs to (denoted \( s(q_i) \)). Sequentially this needs time \( n \log m \) in the worst case.

An important example is: A strip in 2D is partitioned into segments, and queries are points in the strip, see Figure 1. The task is to determine for each query point which segment it belongs to. Note that sorting the points and merging them with the segments would not solve the problem, as our example shows. In case of \( n < m \) we refer to \textit{Multisearch with few queries}, otherwise to \textit{Multisearch with many queries}.

![Fig. 1. Strip with segments and query points. Note, that \( p \) lies left to \( q \) (\( q' \) left to \( p' \)) but \( s(p) \) is right to \( s(q') \) (\( s(q') \) right to \( s(p') \)).](image)

1.2 BSP, BSP* and c-Optimality

The BSP (Bulk-synchronous parallel) model [14] consists of:

- a number of processor/memory components,
- a router that can deliver messages point to point among the processors, and
- a facility to synchronize all processors in barrier style.

A computation on this model proceeds in a succession of \textit{supersteps} separated by synchronisations. For clarity we distinguish between \textit{communication} and \textit{computation supersteps}. In computation supersteps processors perform local computations on data