

Prioritizing Processor Efficiency Over Time Efficiency

Matthew Hudson Walter Xia

October 16, 2016

Abstract

Our paper is *Time-Work Tradeoffs for Parallel Algorithms*, *Spencer[2]*. In this paper, Spencer focuses on several open parallel problems that are affected by the transitive closure bottleneck. For each of the below problems, Spencer describes an algorithm that comes closer to work efficiency than existing approaches that utilize transitive closure. However, this increased work efficiency comes at the cost of an asymptotic increase in the span of each algorithm relative to the time optimal solution. Spencer's motivation for prioritizing work efficiency over time efficiency stems from the fact that existing transitive closure solutions to these problems perform significantly worse than their sequential counterparts when run on only a single processor.

Problems

- Solving Triangular Systems of Linear Equations
- Topological Sort
- Breadth-First Search in Directed Graphs
- Strongly Connected Components
- Single Source Shortest Path

Rather than using a matrix based approach, Spencer utilizes a new data structure called *nearby lists* that serves as the work horse for the subsequent algorithms.

1 INTRODUCTION

Spencer[2] focuses on the problems for which there is no known efficient parallel algorithm that solves them. That is, there is no known parallel algorithm that is able to solve them in work that is within a logarithmic factor of the fastest sequential algorithm, where work is the product of the running time and the processor count. *Spencer[2]* develops parallel algorithms that solves the following problems presented in *Karp-Ramachandran[1]*:

- Directed Spanning Tree
- Breadth-First Search
- Topological Sort
- Cycle Detection for Directed Graphs
- Strong Connected Components
- Single Source Shortest Paths with Non-Negative Edges

His algorithms exhibit a time-work trade-off, that is, the longer they run, the less work they perform. This phenomenon exists due of the fact that certain speculative computations can be eliminated as time progresses, thus reducing work. One algorithm solves topological sort and the other algorithm solves breadth first search. It was mentioned in passing that the breadth first search algorithm can solve directed spanning trees and the topological sort algorithm can solve cycle detection for directed graphs. Furthermore, the breadth first search algorithm can be applied to strongly connected components and extended to single source shortest paths with non-negative edges.

/ Note: We may or may not need these definitions here. */*

We next reproduce three definitions given in *Spencer[2]*:

Definition. A vertex v *reaches* a vertex u iff there is a path from v to u . Conversely, u is *reachable* from v iff there is path from v to u .

Definition. Let $G = (V, E)$ be a graph and let $R \subseteq V$. $G[R] = (R, E')$, where $E' \subseteq R \times R$.

Definition. Two vertices u, v are *identified* by replacing them with a single vertex uv . The outgoing edges of both u and v becomes the outgoing edges of uv and the incoming edges of both u and v become the incoming edges of uv .

2 RESULTS AND TECHNIQUES OF ASSIGNED PAPER

Let $G = (V, E)$, $m = |E|$, and $n = |V|$. The major contributions of *Spencer[2]* are as follows:

- Algorithms that solves topological sort and breadth first search in $O(\frac{n}{\rho} \log^2 \rho)$ time with ρ^3 processors on an EREW PRAM such that $\sqrt{\frac{m}{n}} \leq \rho \leq n$.
- A randomized algorithm that solves strongly connected components in a directed graph in $O(\frac{n}{\rho} \log^2 \rho \log n)$ expected time and $O(n\rho^2 \log^2 p \log n)$ expected work such that $\sqrt{\frac{m}{n}} \leq \rho \leq n$.
- An algorithm that solves single source shortest paths in $O(\frac{n}{\rho} \log n \log(L\rho))$ time and $O(n\rho^2(\log n + \log(L\rho) \log \rho) + m(\log n + \log(L\rho)))$ work such that $\log(L\rho) \leq \rho \leq n$, where L is the length of the longest edge.

2.1 TECHNIQUES

Definition. The *nearby list*, $NL(v)$, of a vertex v consists of ordered pairs $(u, d(v, u))$ where u is a vertex reachable from v and $d(v, u)$ is the distance from v to u .

Definition. A vertex v is *terminal* if $r(v)$, the nearby radius of v , is infinite and $NL(v)$ contains all reachable vertices from v . A vertex v is *totally terminal* if v is terminal and all vertices in $NL(v)$ is terminal.

In solving topological sort, *Spencer[2]* actually solves the reverse topological sort which can be easily converted to a topological sort by replacing $w(v)$ with $n + 1 - w(v)$, where $w(v)$ is the label given by the reverse topological sort. *Karp-Ramachandran[1]* presented an algorithm that can compute this topological sort in $O(\log^2 n)$ time and $O(n^3 \log n)$ work by computing the transitive closure, G^* , of G . *Spencer[2]* introduces an algorithm, *Rtopo*, that performs less work by computing parts of G^* instead of the whole thing. This is achieved by the nearby list data structure, whose size is controlled by the parameter ρ . The smallest $w(v)$ is assigned to the totally terminal vertices, and the nearby lists are updated to expose newly formed totally terminal vertices. The details of which we omit here.

Definition. A vertex v is *known* iff the distance from s to v is at most R_k , the known radius.

In solving breadth first search, *Spencer[2]* employs a very similar algorithm to *Rtopo* called *pbfs*. Again, there exists a parallel algorithm that can solve breadth first search in $O(\log^2 n)$ time and $O(n^3 \log n)$ work by calculating the transitive closure G^* . *pbfs* reduced the work by deleting vertices from G_u , the unknown graph consisting of unknown vertices, once it has been known. All the techniques developed for *Rtopo* are applicable to *pbfs*.

3 SIGNIFICANT RELATED RESULTS

The three most centrally related papers to Spencer we chose are: Karp-Ramachandran, Ullman-Yannakakis, and Paul.

Karp-Ramachandran is a survey of the accumulated theory of parallel algorithms up to the 1990s. This survey focused on the PRAM as its main computational model and starts off with an overview of efficient parallel algorithms on arrays, algebraic expressions, graphs, sorting, etc... After establishing the fundamentals, the survey explores the different variants PRAM models handle concurrent reads and write, in addition provided a brief introduction to other parallel models different from the PRAM. Finally, the survey discussed the class of problem known as NC, problems that have parallel solutions in polylog time and perform polynomial work. It is actually here in which Spencer makes use of Karp-Ramachandran, where the survey listed the aforementioned six graph problems that have no known efficient solution due to the Transitive Closure Bottleneck. We believe that without Karp-Ramachandran, there would be no Spencer, since his entire paper is stemmed from this part of the survey. There is also the fact that Spencer's *Rtopo*'s algorithm depends on an efficient simulation of CREW on EREW to maintain the necessary data structures for its nearby lists.

Ullman-Yannakakis introduces a high-probability parallel algorithm that finds the transitive closure from a single source in $O(n^{e'})$ time with $O(mn^{(1-2e')})$ processors such that $e' \geq n^{(2-3e')}$. Their algorithm also of $e')$ processors such that $m \geq n^{(2-2e')}$. They require $0 \leq e' \leq 1/2$. The probability that their algorithm fails is ac , where a is a positive constant and c is some time multiplier. Their algorithm is the first in which time is

Yannakakis is the fact that they have already established an algorithm exhibiting the time–work tradeoff of probability, and we were interested in seeing a different approach to the same problem we are facing in Spencer.

Paul introduces parallel techniques than search, insert, and delete from 2-3 trees in $O(\log n + \log k)$ time on the EREW, where n is the number of leaves and k is the number of searches, inserts or deletes we want to perform in parallel. He seemed to be the first to design a fast tree algorithm. This results presented by Paul were used by Spencer at several key areas in his algorithms. In particular, Rtopo needed a variation of the search algorithm to find all the totally terminal vertices to be deleted and SSSP required a parallel extension of 2-3 trees to search, insert, and delete many items in parallel. We decided to unpeel some of the details presented in Paul that Spencer decided to abstract away from his paper so that we may have a better understanding of his algorithms.

4 PRESENTATION TOPICS

Here each team member should cite the paper from which they will present technical results in their presentation, and should indicate the exact theorem(s)/lemma(s) whose proof(s) they will present.

5 FURTHER RESEARCH DIRECTIONS

Give an overview of the current open problems and future research directions related to the work in the assigned paper.

REFERENCES

- [1] R. M. Karp and V. Ramachandran. A survey of parallel algorithm for shared-memory machines. *Handbook in Theoretical Computer Science*, pages 35–57, 1990.
- [2] T. H. Spencer. Time-work tradeoffs for parallel algorithms. *J. ACM*, 44(5):742–778, Sept. 1997.