On the Parallel Complexity of Digraph Reachability

Samir Khuller * Uzi Vishkin †

Abstract

We formally show that the directed graph reachability problem can be reduced to several problems using a linear number of processors; hence an efficient parallel algorithm to solve any of these problems would imply an efficient parallel algorithm for the directed graph reachability problem. This formally establishes that all these problems are at least as hard as the $s-t$ reachability problem.

1 Introduction

Many problems are hard to solve *efficiently* in parallel. Despite belonging to the class NC (can be solved in poly-logarithmic time with a polynomial number of processors), they have eluded work-efficient parallel algorithms that run in poly-logarithmic time. One such example is: given a directed graph, is there is a path from s to t (also called the directed graph reachability problem)? The only known poly-logarithmic time algorithms for this problem use $M(n)$ processors on a PRAM, where $M(n)$ is the number of processors required to do matrix multiplication in $O(\log n)$ time (currently, $M(n) = n^{2.376}$[3]). We do not consider such an algorithm to be work-efficient since there exists a linear time serial algorithm for this problem. There are many other problems that also suffer from the same transitive closure bottleneck, and have eluded efficient parallelization. For more on the transitive closure bottleneck see Section 4. 8. 1 in [4]. We formally show that the directed graph reachability problem can be reduced to several such problems using a linear number of processors; hence an efficient parallel algorithm to solve any of these problems would imply an efficient parallel algorithm for the directed graph reachability problem. This establishes that all these problems are at least as hard as the $s-t$ reachability problem. This underscores the importance of solving the directed reachability problem by an efficient parallel algorithm. From another perspective, under the assumption that directed graph reachability is a hard problem, all these other problems are also hard.

2 Description of Problems

In this section we formally define all the problems that are discussed in this note.
Directed Reachability: Given a directed graph $G = (V, E)$ with special vertices s and t, is there a directed path from s to t?

*Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. Research supported by NSF Research Initiation Award CCR-9307462. Part of this work was done while this author was with UMIACS and supported by NSF grant CCR-8906949. E-mail : `samir@cs.umd.edu`.

†Institute for Advanced Computer Studies and Department of Electrical Engineering, University of Maryland, College Park, MD 20742. Also with Computer Science Dept, Tel-Aviv University. Research supported by NSF grant CCR-9111348. E-mail : `vishkin@umiacs.umd.edu`.

2-SAT: Given a formula $\mathcal{F}$ with variables $X_1, \ldots, X_n$, where the formula is written in Conjuctive Normal Form (CNF) such that each clause has at most two literals, is there a satisfying assignment for the formula? A parallel algorithm for this problem was given by Cook and Luby [2].
Augmenting Path: Given an undirected bipartite graph $G = (V, E)$ with some edges in a matching M, is there an augmenting path between two unmatched vertices? (An augmenting path is one that alternates between free and matched edges.)
Linear Context Free Languages: A CFL is said to be *linear* if all productions are of the form $A \to bB$, $A \to Bb$ or $A \to a$. (Uppercase letters denote non-terminals and lower case letters denote terminals.) There is also a given start symbol S. The language generated by the grammar G is the set of all strings (containing no non-terminals) that can be obtained from the start symbol S, by repeated applications of the productions. Given a linear CFL and a string, we would like to know if the string belongs to the language. A parallel algorithm for this problem was given by Atallah et. al. [1].

3 Reductions

We only outline the first reduction from directed reachability to 2-SAT in detail. The rest of the reductions are sketched briefly.

3.1 Reachability to 2-SAT

Given a directed graph $G = (V, E)$, with vertices s and t, we show how to construct a formula $\mathcal{F}$ such that the formula is not satisfiable if and only if there is a path from s to t.

Theorem 3.1 *We can reduce the directed reachability problem to the 2-SAT problem using a linear number of processors.*

Proof. We may assume that there are no incoming edges to s and no outgoing edges from t (these edges can be dropped without affecting the reachability of t from s). Without loss of generality, assume that the vertices are labelled $v_1, \ldots, v_n$ with $s = v_1$ and $t = v_n$.

We create a formula as follows: there are n variables, with X_i corresponding to v_i, for each edge $e = (v_i, v_j)$ we add a clause $(\overline{X_i}, X_j)$. We also add a clause (X_1) and a clause $(\overline{X_n})$.

If the graph G has a path from s to t then we would like to prove that there is no satisfying assignment. This is easy to see as follows. Clearly X_1 has to be `true`. If there is a path $P = (v_1, v_{j_1}), (v_{j_1}, v_{j_2}), \ldots, (v_{j_k}, v_n)$ then since X_1 is `true`, we have to set X_{j_1} to be `true` (because of the clause $(\overline{X_1}, X_{j_1})$). By the same reasoning, X_{j_2} has to be `true`. Eventually, all variables corresponding to vertices reachable from s (including t) have to be `true`. Thus the clause $(\overline{X_n})$ is `false`.

If G has no path from s to t, then we can obtain a satisfying assignment as follows. Set all the variables corresponding to vertices that are reachable from s to be `true` and all the other vertices to be `false`. Thus X_1 is `true` and X_n is `false`. This also satisfies all the clauses.

Observe that the construction of the clauses can be carried out locally since for each edge we need to compute one clause, and only requires m processors where m is the number of edges in G, and $O(1)$ time. □

3.2 Reachability to Augmenting Paths

For each vertex v_i ($i = 2, \ldots, n-1$) in G create two vertices u_i^1 and u_i^2 in G'. Add an edge (u_i^1, u_i^2) to the edge set. Corresponding to $v_1 = s$ and $v_n = t$ create two vertices u_1^2 and u_n^1 respectively. (We can assume that there are no incoming edges to s and no outgoing edges from t.) For each directed edge (v_i, v_j) we add an edge from (u_i^2, u_j^1). All edges of the form (u_i^1, u_i^2) are in the matching M. The only free vertices in the graph G' are u_1^2 and u_n^1. If there is a directed path from s to t then it is easy to see that this path gives rise to an augmenting path in G' from u_1^2 to u_n^1.

Any augmenting path from u_1^2 to u_n^1 in G' can easily be seen to correspond to a directed path in G from s to t. (Intuitively, entering a vertex v_i is the same as reaching u_i^1 by a free edge. Since the next edge has to be in M, we are forced to go to u_i^2 and then exit u_i^2 by leaving on some free edge.) It is also easy to see that this graph is bipartite.

3.3 Reachability to Linear CFL:

We show how to transform the graph G into a CFL with its productions written in *normal* form. For each vertex v_i we introduce a non-terminal A_i. For each edge (v_i, v_j) we introduce a production rule $A_i \rightarrow aA_j$. The start symbol is A_1. Corresponding to the vertex t (v_n) we have the production $A_n \rightarrow aA_n$ and $A_n \rightarrow b$.

Notice that for any vertex v_k that is reachable from v_1 by a path of length ℓ edges, we can use the productions corresponding to these edges to generate the sequence $a^\ell A_k$. If t (v_n) is reachable from s by a path of ℓ' edges, we get the sequence $a^{\ell'} A_n$. By applying the first production for A_n repeatedly, we can get the sequence $a^{n-\ell'}b$ where n is the number of vertices; thus the string $a^n b$ is a string in the Linear CFL. If t is not reachable from s it is easy to see that this string will not be in the CFL.

4 Open Problems

Another problem that has eluded efficient parallelization is the problem of computing the shortest distance between two given nodes in an undirected graph. Is it possible that in our framework one can reduce the directed reachability problem to the shortest path problem? Clearly, with an asymmetric weight function this is easy to do. We would like to do the reduction using a symmetric weight function on the edges.

References

[1] M. J. Atallah, S. R. Kosaraju, L. L. Larmore, G. L. Miller and S. H. Teng, Constructing trees in parallel, *Proc. of* 1^{st} *Symp. on Parallel Algorithms and Architectures*, pp. 421–431, (1989).

[2] S. A. Cook and M. Luby, A simple parallel algorithm for finding a satisfying truth assignment to a 2-CNF formula, *Information Processing Letters*, 27, pp. 141–145, (1988).

[3] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *Journal of Symbolic Computation*, 9, pp. 251–280, (1990).

[4] R. M. Karp and V. L. Ramachandran, Parallel algorithms for shared memory machines, *Handbook of Theoretical Computer Science (Volume A)*, MIT Press/Elsevier, pp. 869–941, (1992)