On Identifying Strongly Connected Components in Parallel*

Lisa K. Fleischer[1], Bruce Hendrickson[2], and Ali Pınar[3]

[1] Industrial Engrg. & Operations Research, Columbia Univ., New York, NY 10027
lisa@ieor.columbia.edu
[2] Parallel Computing Sciences, Sandia National Labs, Albuquerque, NM 87185-1110
bah@cs.sandia.gov
[3] Dept. Computer Science, University of Illinois, Urbana, IL 61801
alipinar@cse.uiuc.edu

Abstract. The standard serial algorithm for strongly connected components is based on depth first search, which is difficult to parallelize. We describe a divide-and-conquer algorithm for this problem which has significantly greater potential for parallelization. For a graph with n vertices in which degrees are bounded by a constant, we show the expected serial running time of our algorithm to be $O(n \log n)$.

1 Introduction

A *strongly connected component* of a directed graph is a maximal subset of vertices containing a directed path from each vertex to all others in the subset. The vertices of any directed graph can be partitioned into a set of disjoint strongly connected components. This decomposition is a fundamental tool in graph theory with applications in compiler analysis, data mining, scientific computing and other areas.

The definitive serial algorithm for identifying strongly connected components is due to Tarjan [15] and is built on a depth first search of the graph. For a graph with m edges, this algorithm runs in the optimal $O(m)$ time, and is widely used in textbooks as an example of the power of depth first search [7].

For large problems, a parallel algorithm for identifying strongly connected components would be useful. One application of particular interest to us is discussed below. Unfortunately, depth first search (DFS) seems to be difficult to parallelize. Reif shows that a restricted version of the problem (lexicographical DFS) is $\mathcal{P}$-Complete [14]. However, Aggarwal and Anderson, and Aggarwal, et al. describe randomized NC algorithms for finding a DFS of undirected and directed graphs, respectively [1, 2]. The expected running time of this latter algorithm is

* This work was funded by the Applied Mathematical Sciences program, U.S. Department of Energy, Office of Energy Research and performed at Sandia, a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. DOE under contract number DE-AC-94AL85000.

$O(\log^7 n)$[1], and it requires an impractical $n^{2.376}$ processors. To our knowledge, the deterministic parallel complexity of DFS for general, directed graphs is an open problem. Chaudhuri and Hagerup studied the problem for acyclic [5], and planar graphs [10], respectively, but our application involves non-planar graphs with cycles, so these results don't help us. More practically, DFS is a difficult operation to parallelize and we are aware of no algorithms or implementations which perform well on large numbers of processors. Consequently, Tarjan's algorithm cannot be used for our problem.

Alternatively, there exist several parallel algorithms for the strongly connected components problem (SCC) that avoid the use of depth first search. Gazit and Miller devised an NC algorithm for SCC, which is based upon matrix-matrix multiplication [9]. This algorithm was improved by Cole and Vishkin [6], but still requires $n^{2.376}$ processors and $O(\log^2 n)$ time. Kao developed a more complicated NC algorithm for planar graphs that requires $O(\log^3 n)$ time and $n/\log n$ processors [11]. More recently, Bader has an efficient parallel implementation of SCC for planar graphs [3] which uses a clever *packed–interval* representation of the boundary of a planar graph. When n is much larger than p the number of processors, Bader's approach scales as $O(n/p)$. But algorithms for planar graphs are insufficient for our needs.

Our interest in the SCC problem is motivated by the discrete ordinates method for modeling radiation transport. Using this methodology, the object to be studied is modeled as a union of polyhedral finite elements. Each element is a vertex in our graph and an edge connects any pair of elements that share a face. The radiation equations are approximated by an angular discretization. For each angle in the discretization, the edges in the graph are directed to align with the angle. The computations associated with an element can be performed if all its predecessors have been completed. Thus, for each angle, the set of computations are sequenced as a topological sort of the directed graph. A problem arises if the topological sort cannot be completed – i.e. the graph has a cycle. If cycles exist, the numerical calculations need to be modified – typically by using old information along one of the edges in the cycle, thereby removing the dependency. So identifying strongly connected components quickly is essential. Since radiation transport calculations are computationally and memory intensive, parallel implementations are necessary for large problems. Also, since the geometry of the grid can change after each timestep for some applications, the SCC problem must be solved in parallel.

Efficient parallel implementations of the topological sort step of the radiation transport problem have been developed for *structured* grids, oriented grids that have no cycles [4, 8]. Some initial attempts to generalize these techniques to unstructured grids are showing promise [12, 13]. It is these latter efforts that motivated our interest in the SCC problem.

[1] Function $f(n) = \Theta(g(n))$ if there exist constants $c_2 \geq c_1 > 0$ and N such that for all $n \geq N$, $c_1 g(n) \leq f(n) \leq c_2 g(n)$. If $f(n) = \Omega(g(n))$, then this just implies the existence of c_1 and N. If $f(n) = O(g(n))$, then c_2 and N exist.

In the next section we describe a simple divide-and-conquer algorithm for finding strongly connected components. In §3 we show that for constant degree graphs our algorithm has an expected serial complexity of $O(n \log n)$. Our approach has good potential for parallelism for two reasons. First, the divide-and-conquer paradigm generates a set of small problems which can be solved independently by separate processors. Second, the basic step in our algorithm is a reachability analysis, which is similar to topological sort in its parallelizability. So we expect the current techniques for parallelizing radiation transport calculations to enable our algorithm to perform well too.

2 A Parallelizable Algorithm for Strongly Connected Components

Before describing our algorithm, we introduce some notation. Let $G = (V, E)$ be a directed graph with vertices V and directed edges E. An edge $(i, j) \in E$ is directed from i to j. We denote the set of strongly connected components of G by $SCC(G)$. Thus $SCC(G)$ is a partition of V. We also use $SCC(G, v)$ to denote the (unique) strongly connected component containing vertex v. We denote by $V \setminus X$ the subset of vertices in V which are not in a subset X. The size of vertex set X is denoted $|X|$.

A vertex v is *reachable* from a vertex u if there is a sequence of directed edges $(u, x_1), (x_1, x_2), \ldots, (x_k, v)$ from u to v. We consider a vertex to be reachable from itself. Given a vertex $v \in V$, the *descendants* of v, $Desc(G, v)$, is the subset of vertices in G which are reachable from v. Similarly, the *predecessors* of v, $Pred(G, v)$, is the subset of vertices from which v is reachable. The set of vertices that is neither reachable from v nor reach v is called the *remainder*, denoted by $Rem(G, v) = V \setminus \{Desc(G, v) \cup Pred(G, v)\}$.

Given a graph $G = (V, E)$ and a subset of vertices $V' \subseteq V$, the *induced subgraph* $G' = (V', E')$ contains all edges of G connecting vertices of V', i.e. $E' = \{(u, v) \in E : u, v \in V'\}$. We will use $\langle V' \rangle = G' = (V', E')$ to denote the subgraph of G induced by vertex set V'. The following Lemma is an immediate consequence of the definitions.

Lemma 1. *Let $G = (V, E)$ be a directed graph, with $v \in V$ a vertex in G. Then*

$$Desc(G, v) \cap Pred(G, v) = SCC(G, v).$$

Lemma 2. *Let G be a graph with vertex v. Any strongly connected component of G is a subset of $Desc(G, v)$, a subset of $Pred(G, v)$ or a subset of $Rem(G, v)$.*

Proof. Let u and w be two vertices of the same strongly connected component in G. By definition, u and w are reachable from each other. The proof involves establishing $u \in Desc(G, v) \iff w \in Desc(G, v)$ and $u \in Pred(G, v) \iff w \in Pred(G, v)$, which then implies $u \in Rem(G, v) \iff w \in Rem(G, v)$. Since the proofs of these two statements are symmetric, we give just the first: If $u \in Desc(G, v)$ then u must be reachable from v. But then w must also be reachable from v, so $w \in Desc(G, v)$. □

With this background, we can present our algorithm which we call DCSC (for Divide-and-Conquer Strong Components). The algorithm is sketched in Fig. 1. The basic idea is to select a random vertex v, which we will call a *pivot* vertex, and find its descendant and predecessor sets. The intersection of these sets is $SCC(G,v)$ by Lemma 1. After this step, the remaining vertices are divided into three sets $Desc(G,v)$, $Pred(G,v)$, and $Rem(G,v)$. By Lemma 2, any additional strongly connected component must be entirely contained within one of these three sets, so we can divide the problem and recurse.

```
$DCSC(G)$
    If $G$ is empty then Return.
    Select $v$ uniformly at random from $V$.
    $SCC \leftarrow Pred(G,v) \cap Desc(G,v)$
    Output $SCC$.
    $DCSC(\langle Pred(G,v) \setminus SCC \rangle)$
    $DCSC(\langle Desc(G,v) \setminus SCC \rangle)$
    $DCSC(\langle Rem(G,v) \rangle)$
```

Fig. 1. A divide-and-conquer algorithm for strongly connected components.

3 Serial Complexity of Algorithm DCSC

To analyze the cost of the recursion, we will need bounds on the expected sizes of the predecessor and descendant sets. The following two results provide such bounds.

Lemma 3. *For a directed graph G, there is a numbering π of the vertices from 1 to n in which the following is true. All elements $u \in Pred(G,v) \setminus Desc(G,v)$ satisfy $\pi(u) < \pi(v)$; and all elements $u \in Desc(G,v) \setminus Pred(G,v)$ satisfy $\pi(u) > \pi(v)$.*

Proof. If G is acyclic, then a topological sort provides a numbering with this property. If G has cycles, then each strongly connected component can be contracted into a single vertex, and the resulting acyclic graph can be numbered via topological sort. Assume a strongly connected component with k vertices was assigned a number j in this ordering. Assign the vertices within the component the numbers $(j, \ldots, j+k-1)$ arbitrarily and increase all subsequent numbers by $k-1$. $\square$

It is important to note that we do not need to construct an ordering with this property; we just need to know that it exists.

Corollary 1. *Given a directed graph G and a vertex numbering π from Lemma 3, then $|Pred(G,v) \setminus SCC(G,v)| < \pi(v)$ and $|Desc(G,v) \setminus SCC(G,v)| \leq n - \pi(v)$ for all vertices v.*

The cost of algorithm DCSC consists of four terms, three from the three recursive invocations and the fourth from the cost of determining the set of predecessors and descendants. For a graph with all degrees bounded by a constant, this last term is linear in the sizes of the predecessor and descendant sets. Let $T(n)$ be the expected runtime of the algorithm on bounded degree graph G with n vertices. For a particular pivot i, let p_i, d_i and r_i represent the sizes of the recursive calls. That is, $p_i = |Pred(G,v) \setminus SCC(G,v)|$, $d_i = |Desc(G,v) \setminus SCC(G,v)|$ and $r_i = |Rem(G,v)|$. If vertex number i is selected as the pivot, then the recursive expression for the run time is

$$T(n) = T(r_i) + T(d_i) + T(p_i) + \Theta(n - r_i). \qquad (1)$$

Clearly, $T(n) = \Omega(n)$, since we eventually must look at all the vertices. Also, in worst case $T(n) = O(n^2)$, since each iteration takes at most linear time and reduces the graph size by at least 1. We show here that the expected behavior of $T(n)$ is $\Theta(n \log n)$. The average case analysis will require summing the cost over all pivot vertices and dividing by n. So for a graph with constant bound on the degrees, the expected runtime, $ET(n)$, of the algorithm is

$$ET(n) = \frac{1}{n}\sum_{i=1}^{n}[T(p_i) + T(d_i) + T(r_i) + \Theta(n - r_i)]. \qquad (2)$$

Theorem 1. *For a graph in which all degrees are bounded by a constant, algorithm DCSC has expected time complexity $O(n \log n)$.*

Proof. We analyze (2) by partitioning the vertices according to their value of r_i. Let $S_1 := \{v | r_{\pi(v)} < n/2\}$ and $S_2 := \{v | r_{\pi(v)} \geq n/2\}$. We analyze each case separately and show that the separate recursions lead to an $O(n \log n)$ expected run time. Thus, the average of these recursions will also.

Case 1: $r_i < \frac{n}{2}$.

Note that Corollary 1 implies the lower and upper bounds: $p_i \leq i-1 \leq p_i + r_i$ and $d_i \leq n - i \leq d_i + r_i$. Since $r_i < n/2$, it follows that $\min\{i, n-i\} \leq r_i + \min\{p_i, d_i\} < \frac{3n}{4}$. By symmetry, it is enough to consider $p_i \leq d_i$. Then we can bound $\min\{d_i, r_i + p_i\}$ from below:

$$\min\{d_i, r_i + p_i\} \geq \min\{i, n/4\}. \qquad (3)$$

By superlinearity of $T(n)$, we have $T(r_i) + T(p_i) + T(d_i) + \Theta(n - r_i) \leq T(r_i + p_i) + T(d_i) + \Theta(n - r_i)$. Using (3), we can bound the contribution of each i by either $T(i-1) + T(n-i) + \Theta(n)$ or by $T(n/4) + T(3n/4) + \Theta(n)$. This latter case, through a well-known analysis, has a solution of $O(n \log n)$. In the former, at worst, all i contributing here lie at the extremes of the interval $[1, n]$. If there are $2q$ of them, their total contribution to (2) is at most $\frac{2}{n}\sum_{i=1}^{q}[T(i-1) + T(n-i) + \Theta(n)]$. Then, an analysis similar to that used for quicksort yields a $O(n \log n)$ recursion. We reproduce this below for completeness.

$$
\begin{aligned}
&\frac{2}{2q}\sum_{i=1}^{q}[T(i-1)+T(n-i)+\Theta(n)] \\
&= \Theta(n) + \frac{1}{q}[\sum_{i=1}^{q-1}(c_1 i \log i) + \sum_{i=n-q+1}^{n}(c_1 i \log i)] \\
&\leq \Theta(n) + \frac{1}{q}[c_1 \log q \sum_{i=1}^{q-1} i + c_1 \log n \sum_{i=n-q+1}^{n} i] \\
&= \Theta(n) + \frac{c_1}{2}[(q-1)\log q + (2n-q+1)\log n] \\
&= c_1 n \log n + [\Theta(n) - \frac{c_1}{2}(q-1)(\log n - \log q)].
\end{aligned}
$$

The expression in brackets in the last inequality is < 0 for large enough choice of c_1.

Case 2: $r_i \geq \frac{n}{2}$.

By superlinearity of $T(n)$, we can rewrite equation (1) as $T(n) \leq T(r_i) + T(d_i + p_i) + \Theta(n - r_i)$. If we let $a = n - r_i$, we can rewrite this as a function of a and n:

$$T_2(a,n) \leq T(n-a) + T(a) + \Theta(a).$$

By our assumptions in this case, we have that $1 \leq a \leq n/2$. We show that this recursion is $O(n \log n)$ by first showing that this holds for $a = 1, n/2$, and then showing that T_2, as a function of a in the range $[1, n-1]$, is convex. Thus, its value in an interval is bounded from above by its values at the endpoints of the interval.

It is easy to see that $T_2(1,n) = \Theta(n)$, and $T_2(n/2,n) = \Theta(n \log n)$. We suppose, by induction, that $T(r) = c_1 n \log n$ for an appropriate constant c_1 for $r < n$, and that the constant in the Θ term is c_2. Thus, the first derivative of $T_2(a,n)$ with respect to a is

$$c_1(\log a - \log(n-a)) + c_2.$$

The second derivative is

$$c_1(\frac{1}{a} + \frac{1}{n-a}),$$

which is positive for $a \in [1, n-1]$. Thus $T_2(a,n)$ is convex for $a \in [1, n/2]$, and hence $T_2(n) = \Theta(n \log n)$ in this case. □

4 Future Work

For the radiation transport application that motivated our interest in this problem, the graphs will often be acyclic, and any strongly connected components will usually be small. For acyclic graphs, a topological sort, using the methodologies being developed for this application, will terminate. By coupling a termination detection protocol to the topological sort, we can use existing parallelization approaches to quickly determine whether or not a cycle exists. Besides quickly excluding graphs which have no cycles, using topological sort as a preprocessing step allows for the discarding of all the visited vertices, reducing the size of the problem that needs to be addressed by our recursive algorithm. With Will McLendon III and Steve Plimpton, we are implementing such a hybrid scheme and will report on its performance in due course.

Acknowledgements

We benefited from general discussions about algorithms for parallel strongly connected components with Steve Plimpton, Will McLendon and David Bader. We are also indebted to Bob Carr, Cindy Phillips, Bill Hart and Sorin Istrail for discussions about the analysis.

References

1. A. Aggarwal and R. J. Anderson, *A random NC algorithm for depth first search*, Combinatorica, 8 (1988), pp. 1–12.
2. A. Aggarwal, R. J. Anderson, and M.-Y. Kao, *Parallel depth-first search in general directed graphs*, SIAM J. Comput., 19 (1990), pp. 397–409.
3. D. A. Bader, *A practical parallel algorithm for cycle detection in partitioned digraphs*, Tech. Rep. Technical Report AHPCC-TR-99-013, Electrical & Computer Eng. Dept., Univ. New Mexico, Albuquerque, NM, 1999.
4. R. S. Baker and K. R. Koch, *An S_n algorithm for the massively parallel CM-200 computer*, Nuclear Science and Engineering, 128 (1998), pp. 312–320.
5. P. Chaudhuri, *Finding and updating depth-first spanning trees of acyclic digraphs in parallel*, The Computer Journal, 33 (1990), pp. 247–251.
6. R. Cole and U. Vishkin, *Faster optimal prefix sums and list ranking*, Information and Computation, 81 (1989), pp. 334–352.
7. T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press and McGraw-Hill, Cambridge, MA, 1990.
8. M. R. Dorr and C. H. Still, *Concurrent source iteration in the solution of 3-dimensional, multigroup discrete ordinates neutron-transport equations*, Nuclear Science and Engineering, 122 (1996), pp. 287–308.
9. H. Gazit and G. L. Miller, *An improved parallel algorithm that computes the BFS numbering of a directed graph*, Inform. Process. Lett., 28 (1988), pp. 61–65.
10. T. Hagerup, *Planar depth-first search in $O(\log n)$ parallel time*, SIAM J. Comput., 19 (1990), pp. 678–704.
11. M.-Y. Kao, *Linear-processor NC algorithms for planar directed graphs I: Strongly connected components*, SIAM J. Comput., 22 (1993), pp. 431–459.
12. S. Pautz. Personal Communication, October 1999.
13. S. Plimpton. Personal Communication, May 1999.
14. J. H. Reif, *Depth-first search is inherently sequential*, Inform. Process. Lett., 20 (1985), pp. 229–234.
15. R. E. Tarjan, *Depth first search and linear graph algorithms*, SIAM J. Comput., 1 (1972), pp. 146–160.