Finding Strongly Connected Components in Parallel in Particle Transport Sweeps *

William McLendon III †
Bruce Hendrickson ‡
Steve Plimpton‡
Lawrence Rauchwerger†

ABSTRACT

Discrete ordinates methods are commonly used to simulate radiation transport for fire or weapons modeling. The computation proceeds by sweeping the flux across a grid. A particular cell cannot be computed until all the cells immediately upwind of it are finished. If the directed dependence graph for the grid cells contains a cycle then sweeping methods will deadlock. This can happen in unstructured grids and time stepped problems where the grid is allowed to deform. In this paper we present a parallel algorithm to detect cycles in the dependence graphs present in these grids as well as an implementation and experimental results on shared and distributed memory machines.

1. INTRODUCTION

Sweeping methods used in radiation transport discretize the radiation field by angle, and flux propagation is computed for a set of discrete directions or ordinates. The computation for each angle is performed by sweeping the flux across a grid, i.e., a finite element mesh commonly used for fluids or shock hydrodynamics modeling. Radiation enters a mesh cell via faces whose outward normals point upwind, and exits through downwind faces. This implies an order of computation on the grid cells which, for a single ordinate direction, is represented as a directed dependence graph.

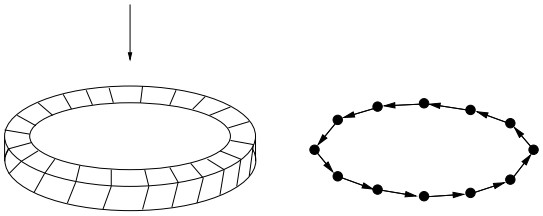

Figure 1: A twisted ring of mesh elements that induces a cycle for the shown angle (left), and a dependence graph for the angle shown (right). A sweeping method will deadlock when it encounters a cycle such as this.

Each of the (typically several hundred) ordinate directions induces an associated dependence graph. Sweeping methods will deadlock if any of the dependence graphs contains a cycle, such as the one in the twisted ring grid shown in Fig. 1. Such situations occur frequently in 3-D unstructured grids and in multi-physics problems where the underlying discretized space (the mesh) deforms over time. To avoid deadlock, cycles in the ordinate dependence graphs must be found and broken before the sweep can be performed.

The number of cycles can be exponential in the number of vertices but the number of *strongly connected components* (SCCs) is at most linear in the number of vertices since a vertex is in at most one SCC. Therefore we are interested in finding all *SCCs* of a directed graph. A *SCC* of a directed graph $G = (V, E)$ is a maximal set of vertices $U \subseteq V$ such that for every pair of vertices u and v in U, we have both $u \rightsquigarrow v$ and $v \rightsquigarrow u$ [1], where $u \rightsquigarrow v$ means there is a directed path from u to v.

Tarjan's classic serial algorithm for detection of SCCs runs linearly with respect to the number of edges and uses depth-first search [7]. However, depth-first search is known to be difficult to parallelize – the special case of lexicographical depth first search is P-Complete [5, 6], which means it is unlikely that a scalable parallel algorithm exists.

2. THE MODIFIED DCSC ALGORITHM

The *Divide-and-Conquer Strong Components* (DCSC) algorithm of Fleischer *et al.* [2] is a recursive, divide–and–conquer approach for finding the SCCs that does not rely on depth-first search. DCSC recursively partitions the dependence graph $G = (V, E)$ so that all SCCs will be entirely contained within a partition. The recursion stops when partitions contain either single vertices or SCCs. The partition-

†Deptartment of Computer Science, Texas A&M University, College Station, TX 77843-3112 {mclendon, rwerger}@cs.tamu.edu
‡Sandia National Labs, Albuquerque, NM 87185-1110 {bahendr, sjplimp}@sandia.gov

*The full version of this work is available in [3]. This work was funded by the U.S. Department of Energy and was performed in part at Sandia National Labs, a multiprogram laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the U.S. DOE under contract number DE-AC-94AL85000 and at Texas A&M University. Research at Texas A&M University was also supported in part by NSF CAREER Award CCR-9734471, NSF Grant ACI-9872126, NSF Grant EIA-9975018, DOE ASCI ASAP Level 2 Grant B347886 and a Hewlett-Packard Equipment Grant.

Algorithm: ModifiedDCSC(G)

```
IF G is empty THEN return
TRIM G in forward direction
IF G is not empty THEN
   TRIM G in backward direction
   Select pivot v from untrimmed vertices
   MARK Pred(G,v) and Succ(G,v) in G
   SCC(G,v) = Pred(G,v) ∩ Succ(G,v)
   DO in parallel:
      ModifiedDCSC(Pred(G,v) − SCC(G,v))
      ModifiedDCSC(Succ(G,v) − SCC(G,v))
      ModifiedDCSC(Rem(G,v))
ENDIF
```

Figure 2: ModifiedDCSC Algorithm.

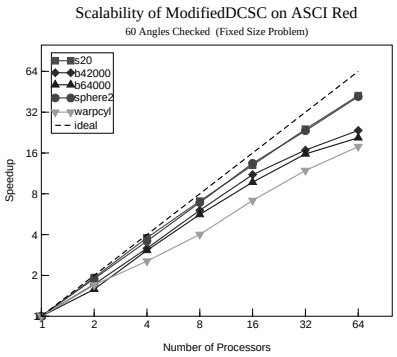

Figure 3: Scalability for various meshes.

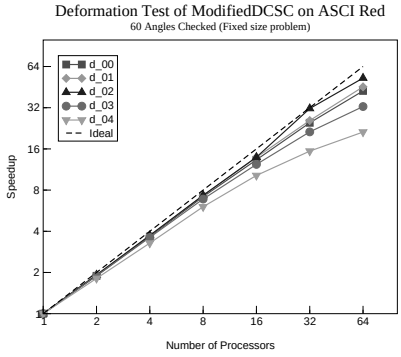

Figure 4: Scalability on meshes of increasing deformation.

ing is based on Lemma 1 [2]:

LEMMA 1. *Let $G = (V, E)$ be a directed graph, with $v \in V$ a vertex in G, and let $Pred(G, v)$ and $Succ(G, v)$ denote the set of predecessors and successors of v in G, respectively. Then, the unique SCC containing v in G, denoted $SCC(G, v)$, is $Pred(G, v) \cap Succ(G, v)$. Moreover, any SCC of G is a subset of $Pred(G, v)$, $Succ(G, v)$, or $Rem(G, v) = V - \{Pred(G, v) \cup Succ(G, v)\}$.*

The expected serial complexity of DCSC is shown to be $O(|V| \log |V|)$ [2].

Our ModifiedDCSC algorithm, outlined in Fig. 2, improves on the basic algorithm by performing a filtering or *trimming* step at the beginning of each iteration which reduces the size of the graph that must be processed. In particular, the *trim* performs forward and reverse topological traversals of G, and all vertices visited in the process are removed from G. No vertices on a cycle, or vertices reachable from a cycle, will be visited by a topological traversal.

Following the trims, DCSC is applied to the remaining portion of the graph. ModifiedDCSC can be called recursively on the 3 new graphs resulting from the DCSC partitioning of the graph.

3. IMPLEMENTATION AND EXPERIMENTAL RESULTS

We have implemented the ModifiedDCSC algorithm in C with MPI [4]. Our code takes as input the finite element grid and a list of ordinates (angles). From this, we generate a dependence graph for each ordinate and store them as a list of distributed graphs.

We take advantage of the discrete ordinates method properties by computing the SCCs of all angles simultaneously and we eliminate redundant work due to paired ordinates.

The systems used for our testing were a 16-Processor HP-V2200 server and the Intel TeraFLOPS (ASCI Red) supercomputer at Sandia National Labs.

We studied the scalability of our algorithm in regards to mesh geometry, mesh dynamics, and message aggregation [3]. All experiments are performed on fixed size data sets.

Figure 3 shows some speedups on ASCI Red on several different meshes of varied shapes and complexity. s20 and Sphere2 have few cycles while warpcyl, b42000, and b64000 have a large percentage of cycles. We see that graphs with fewer cycles scale more effectively.

Multi-physics codes operate on meshes which can be slowly deformed at every time step. As a mesh is deformed, the number of SCCs in the dependence graph can increase dramatically. We generated a 30^3 brick mesh and moved the corner nodes of the cells in some random direction. The magnitude of deformation was increased in increments of 10% of the distance to the nearest corner node in a cell. Meshes d_00–d_04 represent a movement of corner nodes of 0% to 40%. Figure 4 shows the speedups as we progressively deform a mesh on ASCI Red.

The scalability of our heuristic improves as the trimming step can traverse deeper into the graph and involve more processors. When the trims are blocked before they can reach over several hops, fewer processors get involved and thus performance is poorer. In this case our technique relies much more on the more expensive marking step to isolate SCCs.

4. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* M.I.T. Press, Cambridge, MA, 1990.

[2] L. K. Fleischer, B. Hendrickson, and A. Pınar. On identifying strongly connected components in parallel. In *Solving Irregularly Structured Problems in Parallel: 7th Intl. Symp., Irreg. '00*, Vol 1800 of *Lecture Notes in Comp. Sci.*, pp. 505–512. Springer-Verlag, 2000.

[3] William C. McLendon III, B. Hendrickson, S. Plimpton, and L. Rauchwerger. Finding strongly connected components in parallel in particle transport sweeps. Technical Report TR01-005, Texas A&M University, 2001.

[4] William C. McLendon III, B. A. Hendrickson, S. J. Plimpton, and L. Rauchwerger. Identifying strongly connected components in parallel. *Proc. of 10th SIAM Conference on Parallel Processing for Sci. Computing*, March 2001.

[5] Karp and Ramachandran. Parallel algorithms for shared-memory machines. In *Handbook of Theoretical Computer Science, Algorithms and Complexity*, Vol 1. Jan van Leeuwen, Elsevier and MIT Press, 1990.

[6] J. H. Reif. Depth-first search is inherently sequential. *Information Processing Letters*, 20(5):229–234, 1985.

[7] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Computation*, 1:146–160, 1972.