

Identifying Strongly Connected Components in Parallel

William C. McLendon III

Texas A&M University

Bruce Hendrickson

Sandia National Laboratories

Steve Plimpton

Sandia National Laboratories

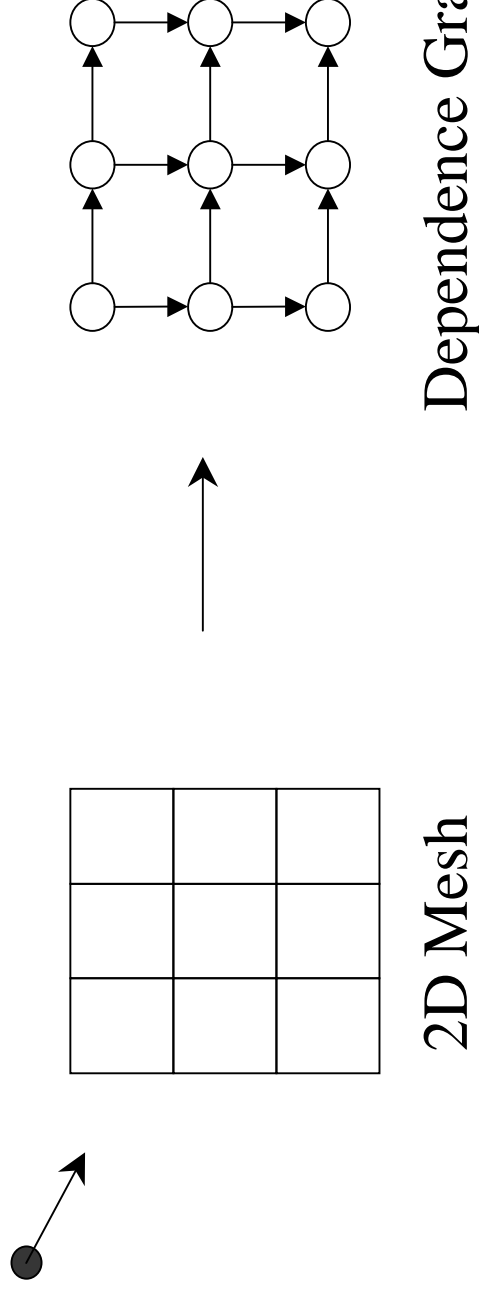
Lawrence Rauchwerger

Texas A&M University

Radiation Transport Motivation

- Sweeping methods are often used for solving many computational problems.
 - Many radiation transport solvers use sweeps
- A sweep imposes an order of computation
 - A mesh cell may not compute until **all** of the cells it depends on are computed.
 - The order of computation may be represented as a dependence graph.
 - The dependence graph must be **acyclic**.

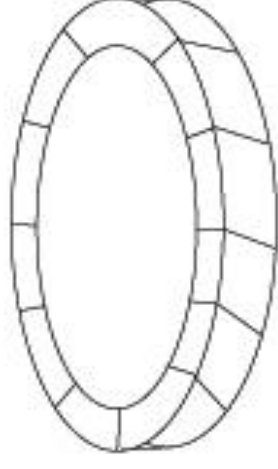
Example Acyclic Dependence Graph



- Example 2D mesh.
- A dependence graph is constructed by comparing the shared face boundary to an ordinate vector.

3D Unstructured Grids May Have Cycles

- The dependence graph of a 3D unstructured mesh is likely to have cycles.
 - Twisted-ring mesh.
 - Time stepped multi-physics codes which deform meshes may introduce cycles.



Strongly Connected Components

- We really need to find the strongly-connected components in the dependence graph.
- A strongly-connected component is:
 - A maximal subset of vertices which for every pair of vertices u, v there is a directed path from u to v and from v to u .
- A single strongly connected component (SCC) can contain many cycles.

Detecting SCCs Optimally

- An optimal algorithm exists to find SCCs in arbitrary graphs due to Tarjan.
 - Tarjan's algorithm relies on Depth-First-Search
- Depth-First-Search (DFS) is not parallel
 - Lexicographical DFS is P-Complete.
 - We cannot use Tarjan's algorithm alone.

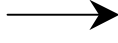
Parallel Application

- We need some method which does not rely on DFS for detecting SCCs in parallel.
- We implemented a modification of the DCSC algorithm from Fleischer, Hendrickson & Pinar
 - DCSC avoids DFS via a series of Breadth-First Searches.
 - Expected serial complexity is $O(n \log n)$
 - Our version adds some extra steps designed to accelerate the detection and reduce the amount of work necessary.
 - We search many angles at once to gain more parallelism.

ModifiedDCSC Terms

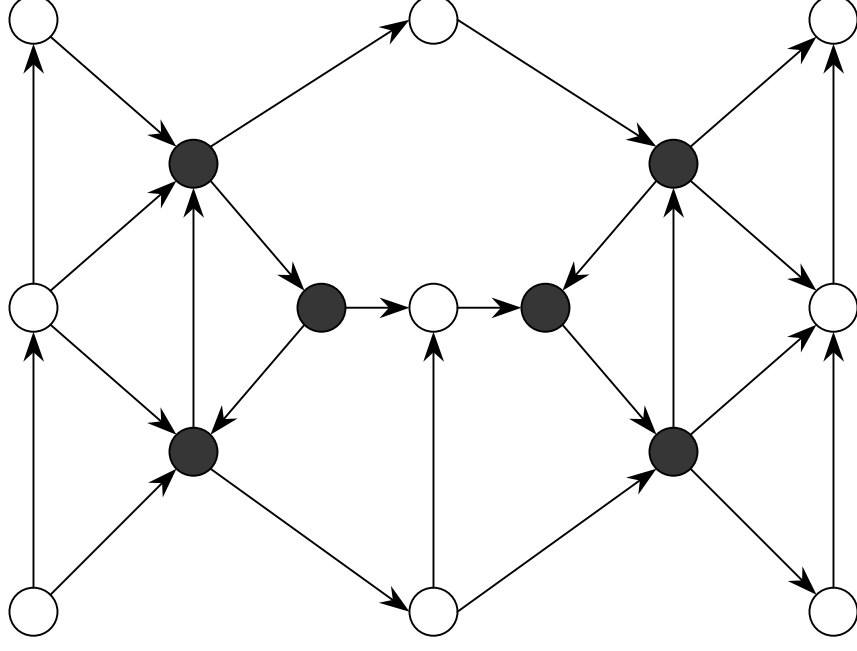
- **TRIM** step
 - Topological traversal which removes nodes as they're visited.
 - Will be blocked by a cycle.
- **MARK** step
 - A node, v , is selected at random from G as a *pivot*.
 - Predecessors are the nodes p in G where a directed path from p to v exists.
 - Successors are the nodes s in G where a directed path from v to s exists.
- Communication in **TRIM** and **MARK** is asynchronous.

Example Graph with Cycles

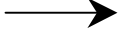


ModifiedDCSC(G)

- **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 the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- **ENDIF**

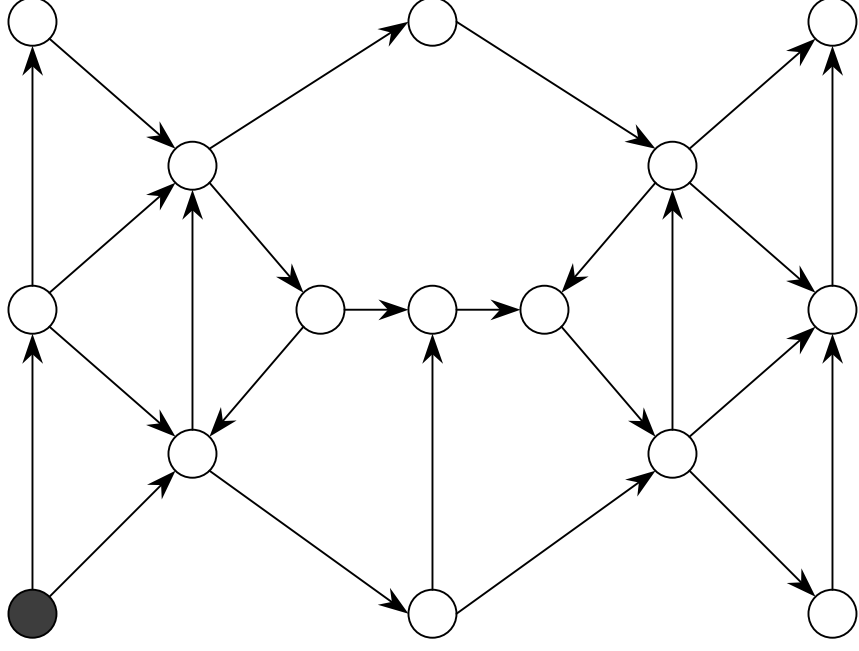


Forward Trim

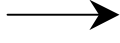


ModifiedDCSC(G)

- 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 the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF

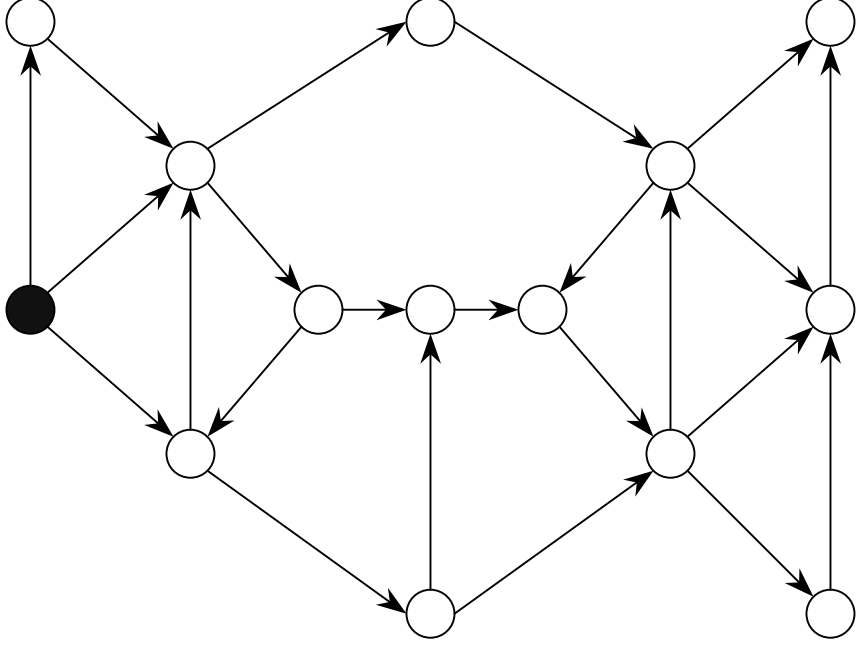


Forward Trim

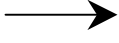


ModifiedDCSC(G)

- 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 the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF

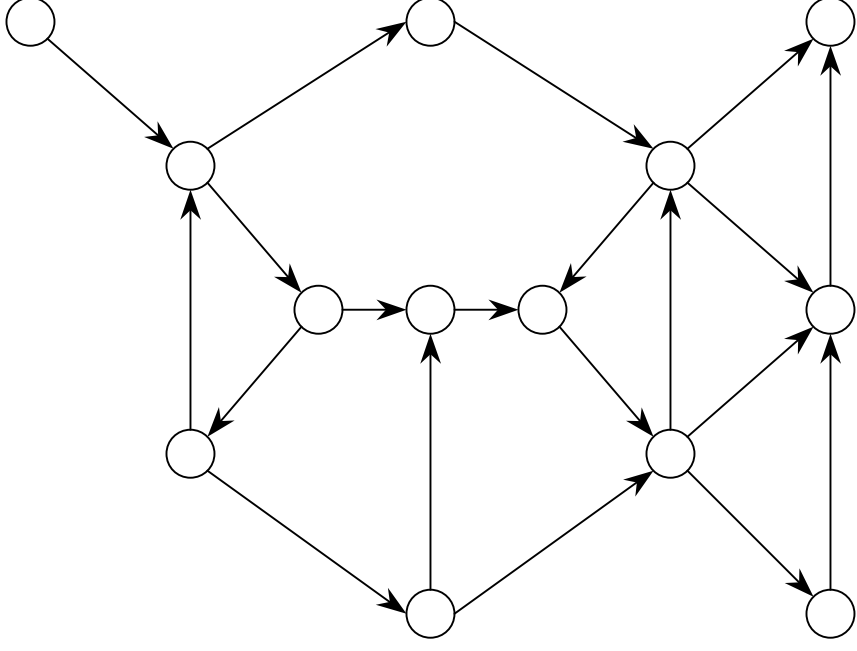


Forward Trim

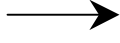


ModifiedDCSC(G)

- 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 the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF

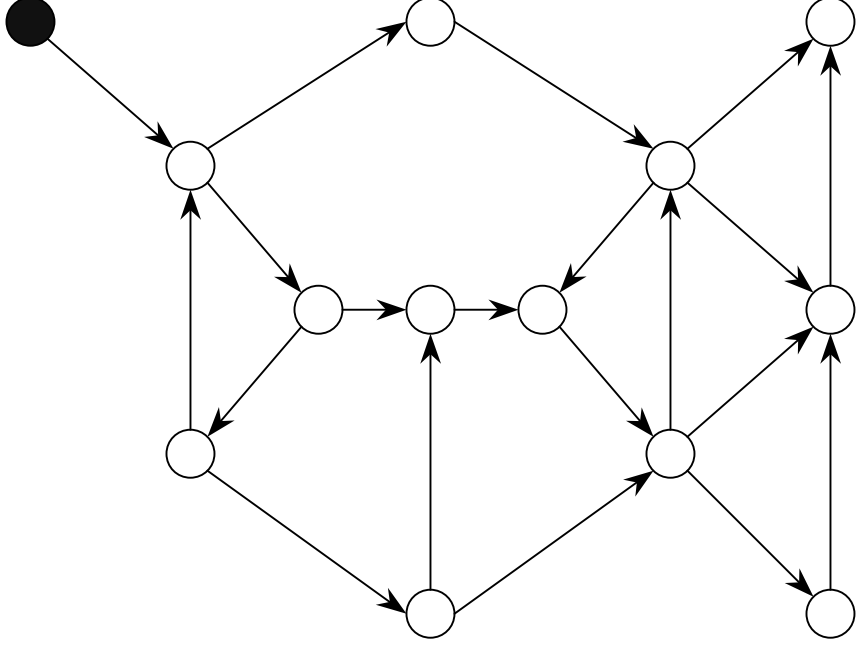


Forward Trim



ModifiedDCSC(G)

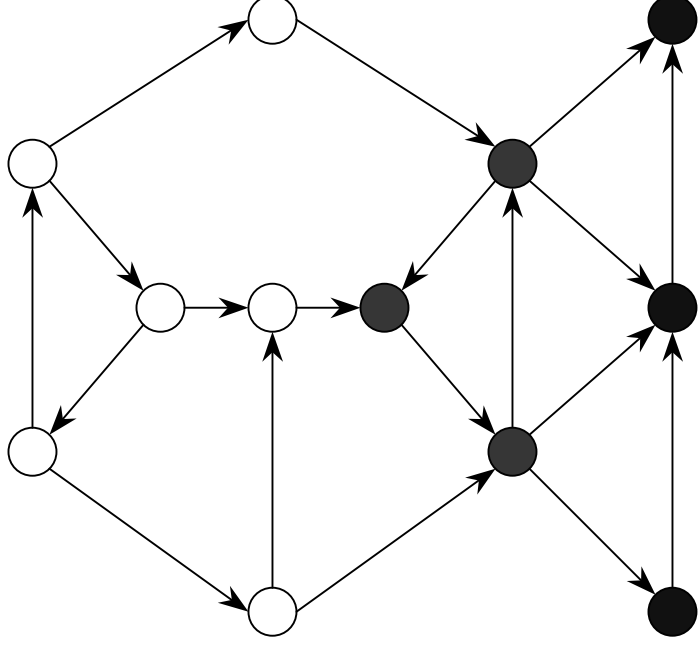
- IF G is empty THEN return
- **TRIM G in forward direction**
- IF G is not empty THEN
- TRIM G in backward direction
- Select pivot v from the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF



Trim in the Backward Direction

ModifiedDCSC(G)

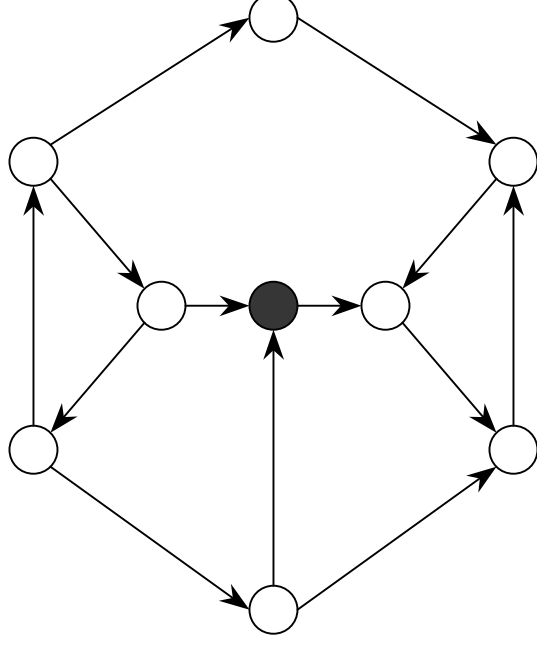
- IF G is empty THEN return
- TRIM G in forward direction
- IF G is not empty THEN
- **TRIM G in backward direction**
- Select pivot v from the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF



Select a Pivot at Random

ModifiedDCSC(G)

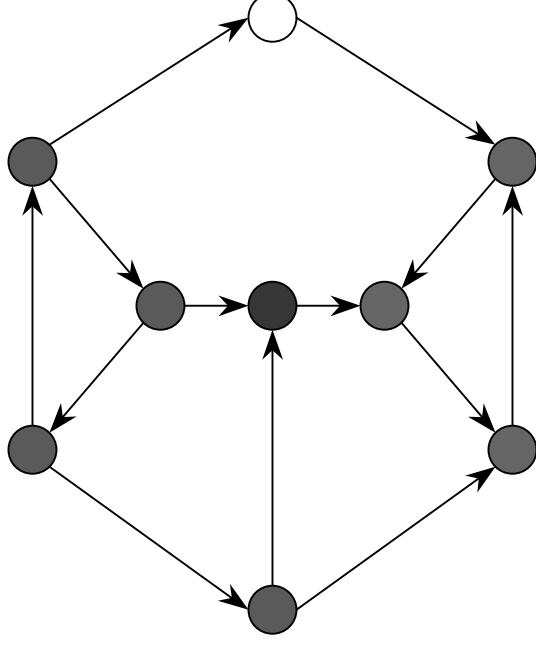
- IF G is empty THEN return
- TRIM G in forward direction
- IF G is not empty THEN
 - TRIM G in backward direction
 - **Select pivot v from the remaining nodes**
 - MARK $Pred(G,v)$ and $Succ(G,v)$ in G
 - $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
 - DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF



Marking the Graph

ModifiedDCSC(G)

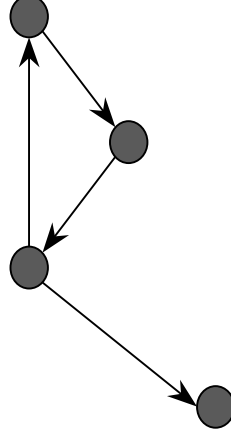
- IF G is empty THEN return
- TRIM G in forward direction
- IF G is not empty THEN
 - TRIM G in backward direction
 - Select pivot v from the remaining nodes
 - **MARK $Pred(G,v)$ and $Succ(G,v)$ in G**
 - $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
 - DO in parallel:
 - ModifiedDCSC($Pred(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Succ(G,v)-SCC(G,v)$)
 - ModifiedDCSC($Rem(G,v)$)
- ENDIF



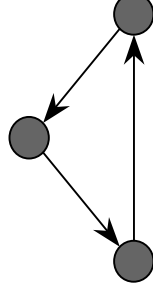
Partitioning the Graph

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 the remaining nodes
- MARK $Pred(G,v)$ and $Succ(G,v)$ in G
- $SCC(G,v) = Pred(G,v) \cap Succ(G,v)$
- DO in parallel:
 - **ModifiedDCSC($Pred(G,v)$ - $SCC(G,v)$)**
 - **ModifiedDCSC($Succ(G,v)$ - $SCC(G,v)$)**
 - **ModifiedDCSC($Rem(G,v)$)**
- ENDIF



Predecessors

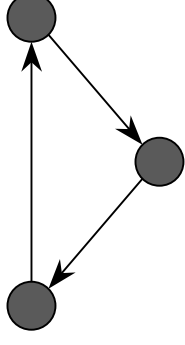
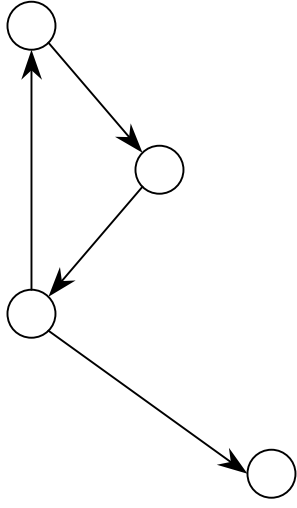


Successors



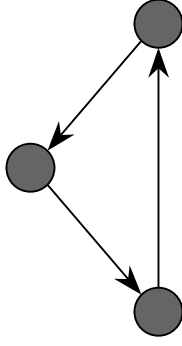
Remainder

Recurse on Every Partition



ModifiedDCSC finds a SCC when the pivot is *part of the SCC*.

Predecessors, successors, and remainder sets do not share cycles.



We find the SCCs with recursive calls to ModifiedDCSC on each subgraph.

Implementation

- ModifiedDCSC is implemented using C and MPI.
- Developed jointly at Sandia National Laboratories and at Texas A&M University.
- Tested on ASCI Red, C-Plant, and on HPUX systems.

Optimizations for Rad. Transport

- Detect SCCs for all angles simultaneously.
- Pairs of opposite ordinates will have same SCCs
 - Opposite ordinates are same graph with edges flipped, this preserves the SCCs.
 - Only search for SCCs in one of each angle pair.
 - Balance these ordinates among octants

Measurements

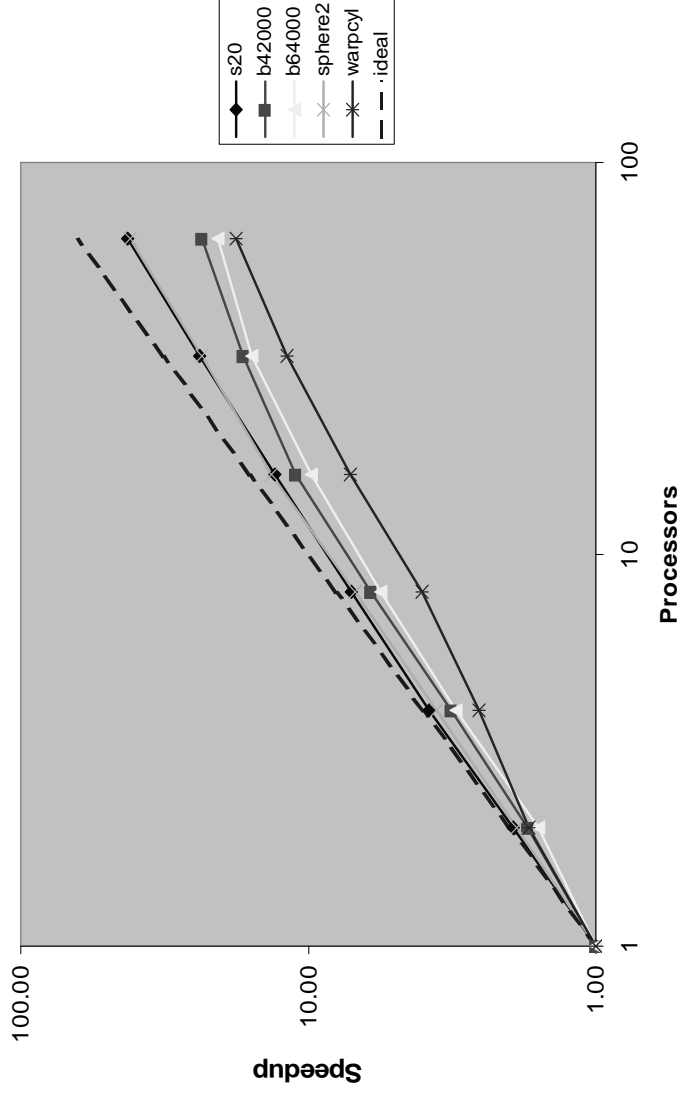
- Tests were run on the following platforms:
 - HP V-Class Server with 16 Processors.
 - ASCI Red at Sandia National Labs.
- ModifiedDCSC is input sensitive.
 - Performance can vary greatly from one mesh to another.
 - We tried to run a varied set of problems on it to attempt to characterize some of the performance characteristics of this heuristic.

Measurements (2)

- Experiments:
 - Varied mesh types.
 - Some realistic meshes from Sandia National Labs.
 - Varied deformation of nodes in a brick.
 - 30x30x30 brick with the corner nodes deformed.
 - Varied amounts of message aggregation.
 - Vary how large the messages must be before they are sent.

Varied Meshes on ASCI Red

Speedup of ModifiedDCSC on ASCI Red



Mesh	Nodes	Angles	Num. SCC	Description
s20	43984	60	4	Area around a submarine hull. Few, localized cycles
sphere2	36712	60	1	Solid sphere.
b42000	42875	60	4420	Brick with many, evenly distributed cycles
b64000	64000	60	17437	Larger version of b42000
warpcyl	28000	60	280	Warped cylinder with stacked concentric rings

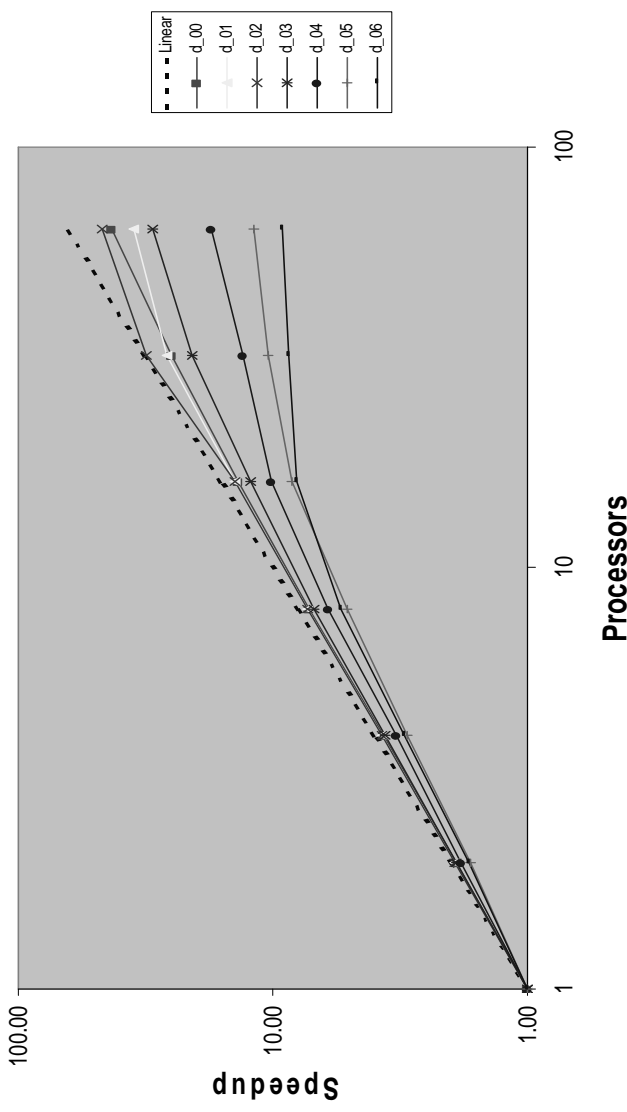
Deformed Mesh Example

Mesh	Deform Magnitude	# SCCs	Execution Time on ASCII Red			
			1	2	4	8
D_00	0%	0	16.55	8.66	4.49	2.29
D_01	10%	0	20.90	10.83	5.57	2.83
D_02	20%	0	20.91	10.83	5.58	2.83
D_03	30%	70	23.39	12.30	6.46	3.37
D_04	40%	899	28.02	15.41	8.46	4.51
D_05	50%	2701	31.48	18.51	10.21	5.61
D_06	60%	4825	34.31	20.31	10.97	5.87
D_07	70%	7120	38.88	22.78	12.44	6.82

As mesh elements are progressively deformed, there are more cycles!

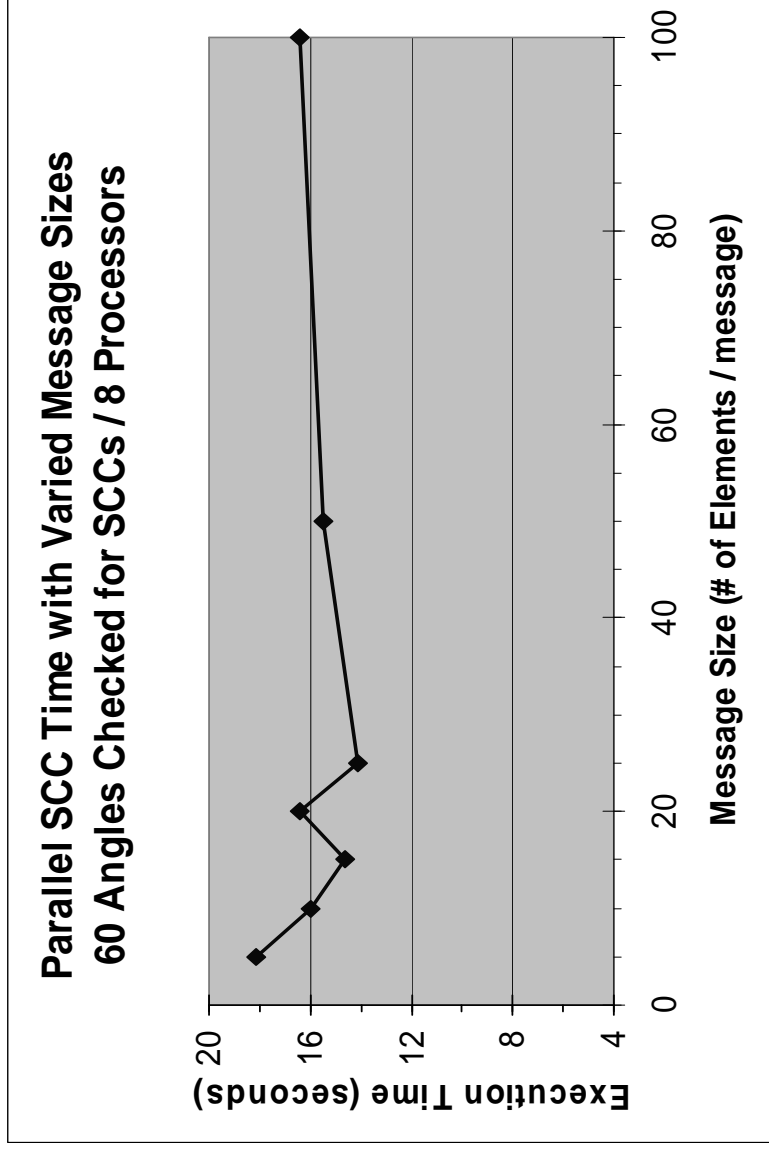
- Performance remains good for problems with few cycles.
- Meshes used for transport sweeps are expected to have relatively few cycles.

Mesh Deformation Speedups on ASCII Red



Message Aggregation

Results from varying the number of nodes per message during the *trim* and *mark* steps.



Message Aggregation Test on b42000	
Message Size	Execution Time (s)
5	18.16
10	15.95
15	14.65
20	16.38
25	14.14
50	15.49
100	16.42

frequent/small messages
vs. infrequent/large
messages.

Conclusions (1)

- We have implemented a parallel strongly connected component heuristic for detecting SCCs in unstructured grid meshes.
- Tested ModifiedDCSC on a variety of meshes showing the performance characteristics and input sensitivity.

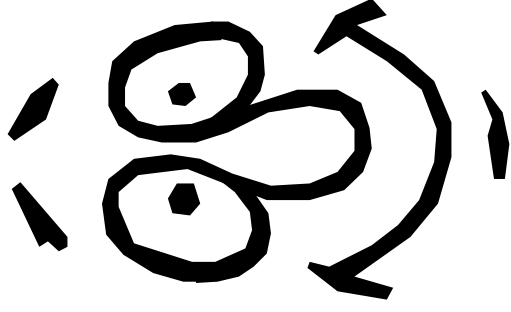
Conclusions (2)

- We expect ModifiedDCSC to perform well for dependence graphs of meshes used for radiation transport sweeps.
 - Relatively few numbers of cycles are expected in these graphs.
 - Execution time is much less than the time for a transport sweep calculation.

Future Plans

- Integration into SNL's ALEGRA multiphysics code.
- Implementation with STAPL.

End of Presentation



WWW Pages

PARASOL: <http://www.cs.tamu.edu/research/parasol/>

William McLendon: <http://www.cs.tamu.edu/people/mclendon/>

Lawrence Rauchwerger: <http://www.cs.tamu.edu/faculty/rwenger/>