**AgES-W Parallelization Algorithms**

Dan Elliott

**Background**

AgES-W delineates watersheds into two types of hydrological units: land units called HRUs and sections of streams and rivers called stream reaches. A core part of AgES-W calculates properties of Hydrological Response Units (HRUs). Because HRUs are a part of a watershed, flow may be routed from HRUs to other HRUs as well as to stream reaches. Once flow is routed to a stream reach, it is routed to other stream reaches until it arrives at the outlet of the watershed. The order in which this flow routing occurs forms a flow topology.

**Graph Theory Terminology**

Flow topologies may be best understood in terms of graph theory. A flow graph is a mathematical graph which consists in a set of vertices that represent hydrological units and a set of directed edges which represent flow from one vertex to another. A graph G is typically represented as G = (V, E) where V is the set of vertices and E is the set of edges.

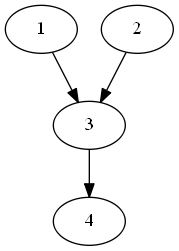


Figure 1. Flow graph G.

For example, let G = (V,E) where V = {1,2,3,4} and E = {(1,3),(2,3),(3,4)}. This means that there are HRUs 1, 2, 3, and 4. There is flow from 1 to 3, 2 to 3, and 3 to 4. During an execution of AgES-W, an HRU must receive flow from all contributing HRUs before it may route flow to other (receiving) HRUs. This constraint is called a precedence constraint. As we can see from the above example, HRUs 1 and 2 can immediately route flow to HRU 3. However, HRU 3 cannot route to HRU 4 until it has received flow from HRUs 1 and 2. In general, if the graph has an arbitrary edge (i,j) then j is said to be dependent on i.

AgES-W considers a flow topology to be valid if it does not contain a cycle or loop. The existence of a cycle makes it impossible to obey precedence constraints.

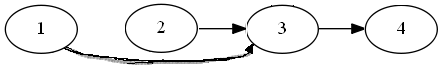


Figure 2. Topological sort of G.

A result of graph theory is that any valid flow topology can be placed in a topological sort. A topological sort is an ordered list of vertices. For any vertex in the list, any of its flow will route to a vertex further in the list. For instance, the lists [1,2,3,4] and [2,1,3,4] are both topological sorts of the example graph G. This ordering ensures that when vertices representing hydrological units are processed sequentially, precedence constraints are obeyed.

A source vertex is a vertex which is not dependent upon any other vertex in the graph. A sink vertex (outlet) is a vertex that doesn’t have any outgoing edges. In the above graph G, vertices 1 and 2 are sources, while vertex 4 is a sink. Every vertex in a flow topology has a dependency depth. Source vertices have a dependency depth of 0 because they aren’t dependent upon any other vertices. The dependency depth value of an arbitrary non-source vertex i is the length of the longest path from a source to i. In the example above, the depth of vertices 1 and 2 are 0, the depth of vertex 3 is 1, and the depth of vertex 4 is 2.

If two vertices u, v have an edge to a vertex z then the vertices u and v are said to be siblings. In our example graph G, vertices 1 and 2 are siblings because they both have an edge to vertex 3.

**Parallelization Algorithms**

These algorithms construct a static schedule of tasks from a set of HRUs.

**Sequential**

The HRUs are placed into a topological sort and are encapsulated into a list of tasks. The scheduled time unit for a task is its index in the list.

**Layer**

The HRUs are assigned their proper dependency depth. Each task is assigned a time unit equal to the dependency depth of the HRU it encapsulates.

**Tango**

The HRUs are placed into a topological sort and are encapsulated into a list of tasks. Each HRU is assigned its proper dependency depth. Tango iterates through the topologically sorted list, building a list of HRUs which obey both precedence constraints and sibling constraints. The list of HRUs are created from the topological sort on iteration i are added to schedule at time unit i. Then these HRUs are removed from the topological sorted list. This process is repeated until the topological sorted list is empty.

**Forest**

Forest parallelization works by partitioning HRUs into disjoint sets of the same type. A partition P of HRUs may be called disjoint if for any HRU h in P, if h routes flow to another HRU, it routes flow to another HRU in P. The stream reach entities form a single tree within AgES-W so there is only a single partition of stream reaches (explain further or differently disjoint and partition not sure if these are known in reference to graphs). When a partition is processed, its HRUs are processed sequentially. The insight is that because a partition won’t route to another partition, each partition can be processed in parallel with respect to all others.

Psuedocode:

In order to form a partition, one performs the following steps on the routing graph G of HRUs.

Let G’ be an undirected graph constructed by causing G’s directed edges to become undirected.

Mark each HRU h in G’ as unexplored.

Define function DFS(G’:graph, h:HRU, parition:list) to be DFS on G’ where all nodes explored in DFS are added to partition.

PartitionSet = [] // empty list of paritions

For HRU h in G’:

If h is unexplored:

partition = [] //empty list of HRUs

DFS(G’,h,partition)

TopologicalSort(partition)

PartitionSet.add(partition)

END Pseudocode

**Experimental Evaluation**

**Results**

**Conclusions and Further Work**

**Appendix**

**Computing Terminology**

A task encapsulates data and its computational work. In AgES-W, task data is a hydrological unit. Additionally, these tasks are assumed to be homogenous, meaning that all tasks of the same type take the same amount of execution time and other computational resources. A static schedule is an assignment of positive integer-values to tasks. The value assigned to a task is called a time unit. All tasks of the same time unit can be executed in parallel. After all tasks of time unit t have been executed, tasks of time unit (t+1) may be executed. The length of a schedule is equal to the maximum time unit value assigned by the schedule. A sibling constraint says that siblings must be processed in different time units.