Directed Acyclic Graph Conversion to Tree with Two Edge Types

John Vivian

June 2015

Due to the adoption of the Directed Acyclic Graph (DAG) as a standard in many workflows (e.g. Common Workflow Language), it is useful to have an algorithm capable of converting DAGs into trees, which is the data structure jobTree uses to model job hierarchy. Since standard trees, with one type of edge, are incapable of storing the complexity that can arise from a DAG, we will use a special type of tree that has 2 types of edges and is used by **jobTree**, a scheduling system used by UCSC's Computational Genomics Lab.

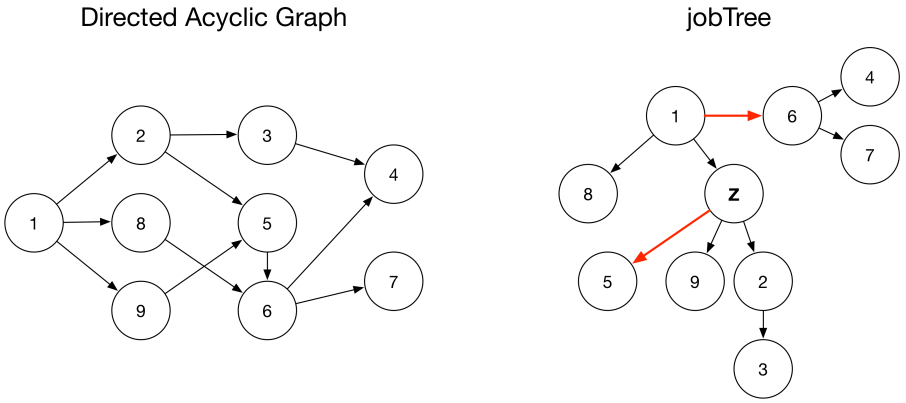

Figure 1: An example of a DAG reshaped as a tree. The black arrows in the tree correspond to 'child' jobs (collectively children) and the red arrows correspond to 'follow-on' jobs, which are only run after all of a given node's children and their descendants have run. **Z** represents a pseudo-node that has no function outside of scheduling jobs.

1 Algorithm

1.1 Setup

if $\exists > 1$ *source node* **then**
  Create new source node ***S***;
  for *node* $\in$ *nodes* **do**
    (***S***, *node*)
  end
end

1.2 Creating Pseudonodes to Remove Multiple Parents

while $\exists \geq 1$ *node with multiple parents* **do**
  Let **X** be a node with multiple parents whose predecessors each have only one parent;
  Let **Y'** be the most recent common ancestor of **X**'s parents.;
  if ***Y' has a follow-on*** **then**
    Y is the node on a path from **Y'** to **X** that is connected to **Y'** by a sequence of follow-on edges.;
  else
    Y' is **Y**
  end
  Make new child of **Y**, **Z**;
  for *each edge incident to* ***Y*** *that is on the path to* ***Z*** **do**
    Delete (**Y**, incident node);
    Attach (**Z**, incident node);
  end
  for *each incoming edge of* ***X*** **do**
    Delete edge
  end
  Add follow-on edge (**Z**, **X**)
end

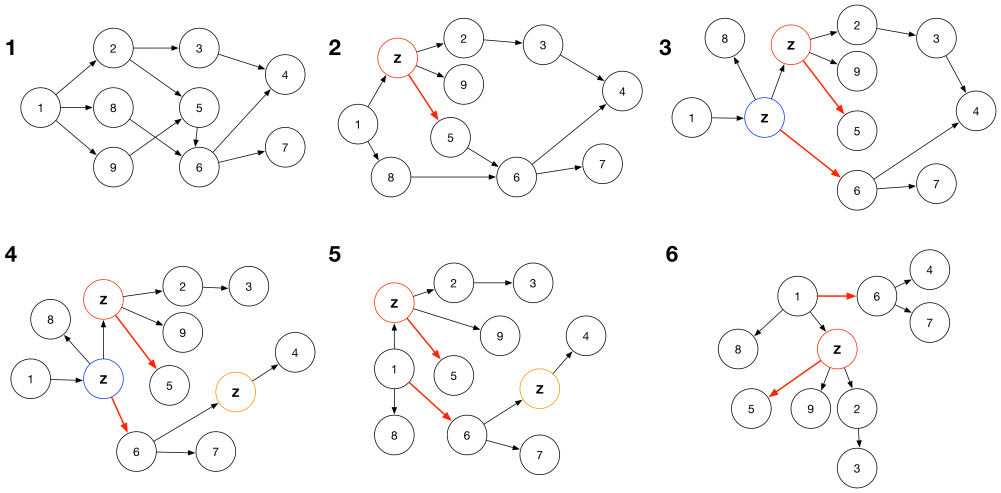

Figure 2: DAG to Tree Conversion