Graph Embeddability with Constrained Reachability
Matthew Chu
Faculty Advisors: Sampath Kannan, Sanjeev Khanna

Abstract:
- Given a transitive directed reachability graph $G$ and a target structure $S$, we are interested in whether or not we can create an embedding of $G$ in $S$.
- Definition: A Steiner point is a vertex $s$ in $S$ that does not correspond to any vertex $v$ in $G$.

Lines:
- A saw-tooth graph with two terminals is defined inductively as follows:
  1. Any total ordering graph is a saw-tooth graph with the same set of terminals.
  2. If $G_1$ is a saw-tooth graph and $G_2$ is a total-ordering graph, then the graph obtained by identifying one terminal of $(in(out))$-degree 0 in $G_1$ with one terminal of $(in(out))$-degree 0 in $G_2$ is a saw-tooth graph. The terminals of this graph are the two remaining terminals of $G_1$ and $G_2$.
- The class of saw-tooth graphs is exactly the class of graphs embeddable on a line without Steiner points.

Trees Without Steiner Points:
- A tree order is defined inductively as follows:
  1. Any total order is by itself a tree order.
  2. If $T$ is a tree order and $S$ is a total order, they can be joined to make a larger tree order by identifying exactly one of the end points of $S$ with a vertex in $T$.
- The class of tree orders is exactly the class of graphs embeddable in a tree without Steiner points.
- Algorithm for simplifying a graph:
  1. Transitively reduce the graph. This is done by removing edges from a vertex $a$ to a vertex $c$ if there exists a vertex $b$ such that there is an edge from $a$ to $b$ and an edge from $b$ to $c$.
  2. Remove all vertices that would be in the middle of a total order. This is done in the following manner: If there is an edge from a vertex $a$ to a vertex $b$, an edge from $b$ to a vertex $c$, and no other edges involving $b$, then remove vertex $b$ and add an edge from $a$ to $c$.
  3. A reachability graph can be embedded into a tree without Steiner points if and only if its simplified graph is connected and does not have any undirected cycles.

Trees With Steiner Points:
- In this case, an alternating 4 cycle can be embedded, but the only way to embed it is in the form of a Steiner embedded alternating 4 cycle.

Complete Layered Grids Without Steiner Points:
- Without Steiner points, the only thing that could be embedded is a $p$ by $q$ complete grid, where $pq$ is the number of vertices in the original reachability graph.
- Algorithm for detecting whether a reachability graph can be embedded:
  1. Transitively reduce the graph.
  2. Identify all vertices of degree 0 (there should be only 1, if not this cannot be embedded) or degree 1. Together, these should form two total orders. If this is the first pass, define $p$ and $q$ by the lengths of the total orders. If this is not the first pass, the required lengths of the total orders will be passed in and if the total orders are not of the required lengths, return that this is not embeddable.
  3. Recurse on the graph minus the vertices identified in step 1 with the required total order lengths 1 less than the current required total order lengths.
  4. Connect the two total orders identified in step 1 as the leftmost column and topmost row and ensure that all reachability constraints with regard to these vertices are satisfied (Note: if $p = q$, this could require switching which total order is put vertically and which is put horizontally).

Complete Layered Grid With Steiner Points:
- An antichain in a reachability graph $G(V,E)$ is a subset $U$ in $V$ such that for all $u, v$ in $U$, $u$ cannot reach $v$ in $G$.
- Path Condition: For three vertices $a$, $b$, and $c$ in an antichain, if there is an undirected path from $b$ to $c$ where none of the vertices on the path are reachable from $a$ or reach $a$, then $a$ is not between $b$ and $c$.
- Private Vertex Condition: For three vertices $a$, $b$, and $c$ in an antichain, if there is a vertex $d$ that is reachable from all 3 and there is a vertex $e$ that is reachable from $a$, not reachable from $b$ and $c$, and cannot reach $d$, then $a$ is not between $b$ and $c$.
- For an antichain containing $a$, $b$, and $c$, $b$ is between $a$ and $c$ if:
  1. $a$ is not between $b$ and $c$ by the Path or Private Vertex Conditions.
  2. $c$ is not between $b$ and $a$ by the Path or Private Vertex Conditions.
- In the embedding of an antichain, there is an induced total order by looking at the vertices going from south to north. We define $a < b$ if $b$ is north of $a$. Furthermore, if $b$ is between $a$ and $c$, then in any embedding, $a < b < c$.
- Algorithm for creating an ordering of an antichain:
  1. Pick any vertex $R$ that is not between any other vertices in this antichain. If there is no such vertex, fail.
  2. Set $R$ as the next lowest vertex in the ordering.
  3. Refine the existing groups as follows: Break each group into subgroups based on which vertices they can reach that $R$ can also reach. If this is the first pass, define $p$ and $q$ by the lengths of the total orders. If this is not the first pass, the required lengths of the total orders will be passed in and if the total orders are not of the required lengths, return that this is not embeddable.
  4. Recurse on the graph minus the vertices identified in step 1 with the required total order lengths 1 less than the current required total order lengths.
  5. Connect the two total orders identified in step 1 as the leftmost column and topmost row and ensure that all reachability constraints with regard to these vertices are satisfied (Note: if $p = q$, this could require switching which total order is put vertically and which is put horizontally).
  6. From the first group (call it $G$), if there is only 1 vertex, go to step 7. If not, go to step 8. Call the group of vertices that each group can reach that $R$ can also reach. If the sequence is not monotonically decreasing sequence with respect to subset, then fail.
  7. Set the only vertex in $G$ as $R$ and return to step 2.
  8. For every vertex $T$ in $G$, find the set of vertices that $T$ can reach and that cannot reach any vertices in $H$. Organize these in an ascending sequence, and if the resulting sequence is not monotonically increasing, fail. Among the vertices for which the group constructed is minimal, choose any vertex that is not between any two unplaced vertices. Call this chosen vertex $R$ and return to step 2.
  9. If the input is a bipartite graph, the above algorithm can be used to determine whether or not there is an embedding.