Homework 11

Adam Karl

November 3, 2020

1 Problem 34 (6 points)

1.1 Motivation

Let this problem be called DisjointDirectedPaths. If we are able to reduce 3SAT to DisjointDirectedPaths using polynomial time transformations such that 3SAT is satisfied if and only if DisjointDirectedPaths returns true, then since 3SAT is a known NP-hard problem DisjointDirectedPaths must be NP-hard as well.

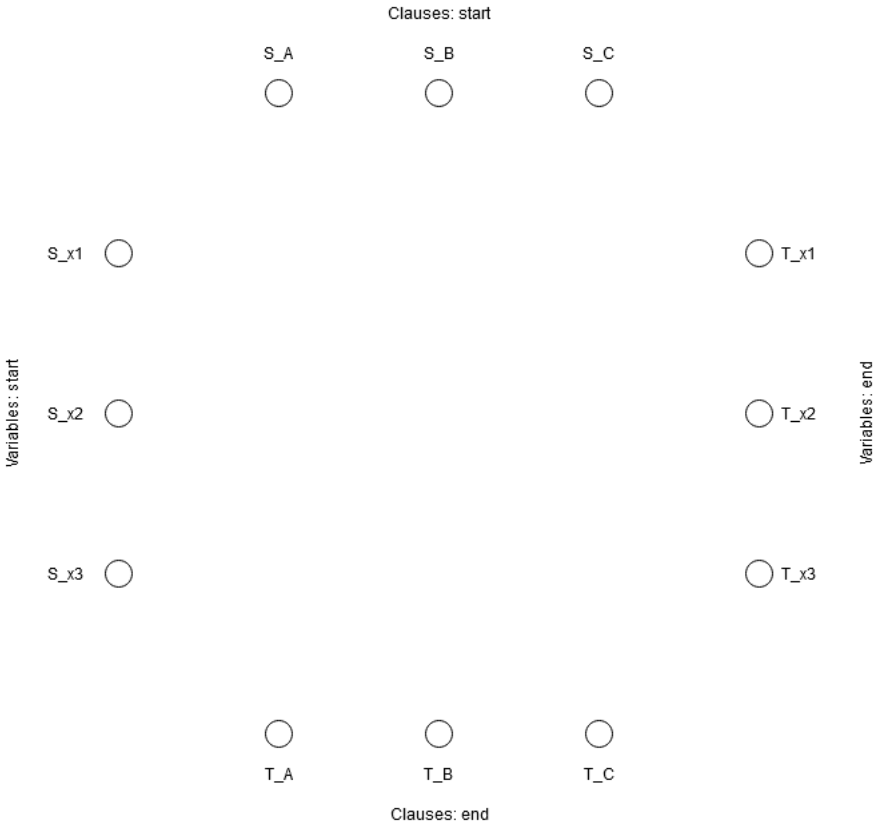
1.2 Transformation Description

Example 3SAT input : $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

To solve 3SAT using an algorithm for DisjointDirectedPaths, do:

For each variable, create an s-t vertex pair. In the example diagram they are aligned horizontally.

For each clause, create an s-t vertex pair. For the diagram, clauses are named A, B, and C, and are aligned vertically.



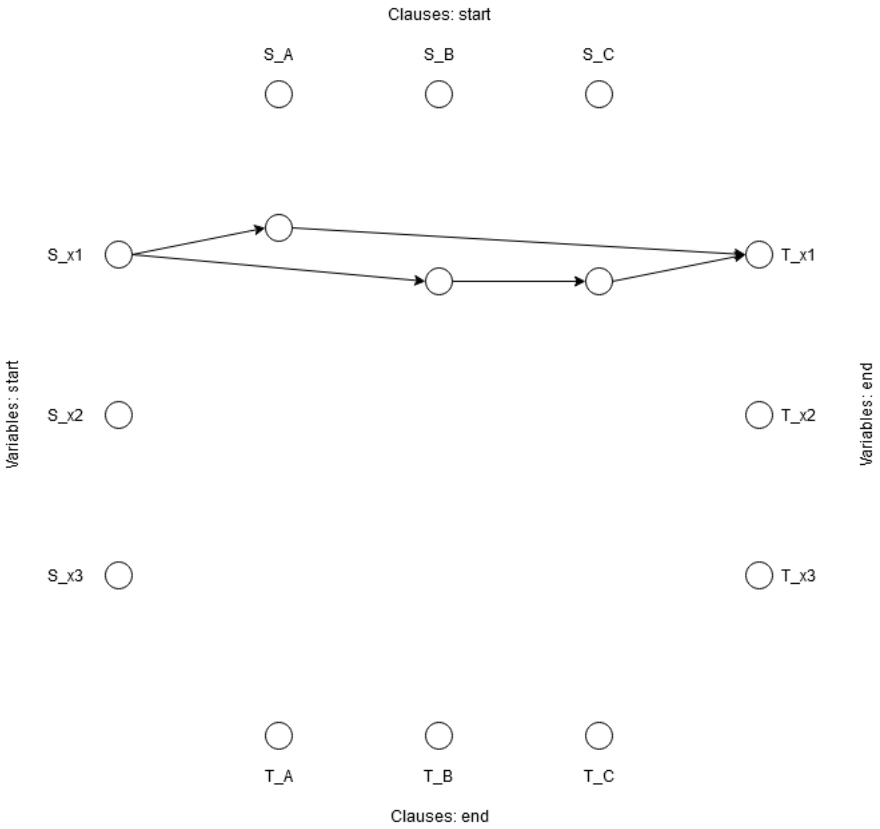
all s-t pairs defined

For each variable, find all clauses it appears in and create a vertex for each time it appears. Create a path starting at the variable's s vertex, through each vertex where that variable is not

negated for that clause, and ending at the variable's t-vertex. This is depicted as the "high road" in the example, and only goes through the vertex corresponding to clause A because that is the only clause where x_1 is not negated.

Then create another path from s and ending at t, passing through all the vertices corresponding to clauses where the variable is negated in the corresponding clause. This is the "low road" in the example diagram.

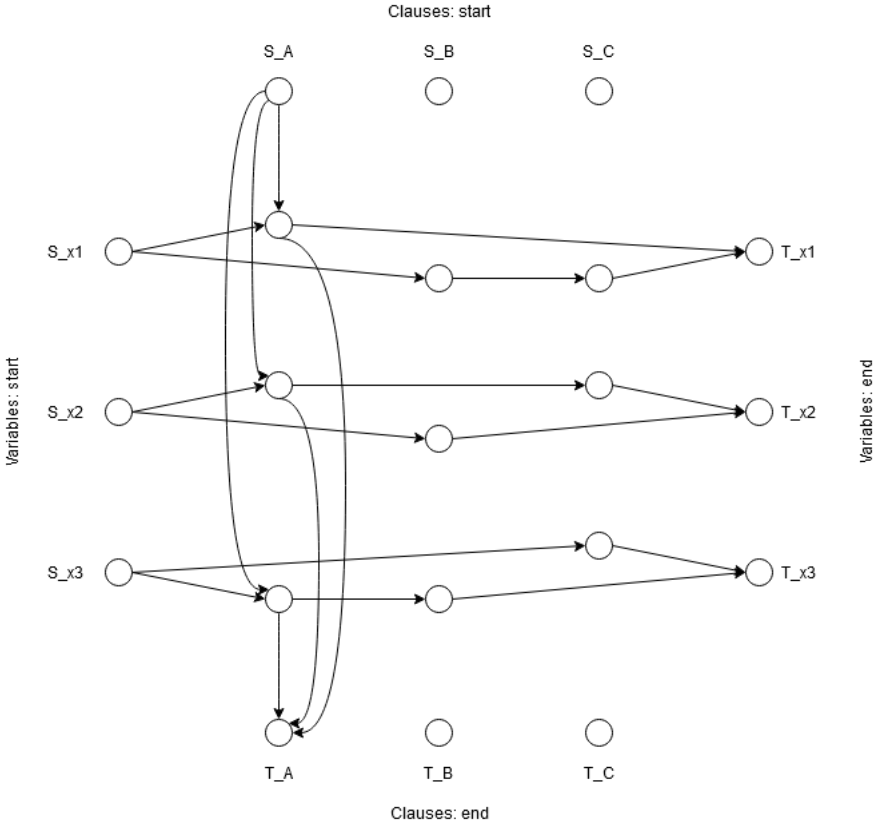
input: $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$



one variable (x_1) added

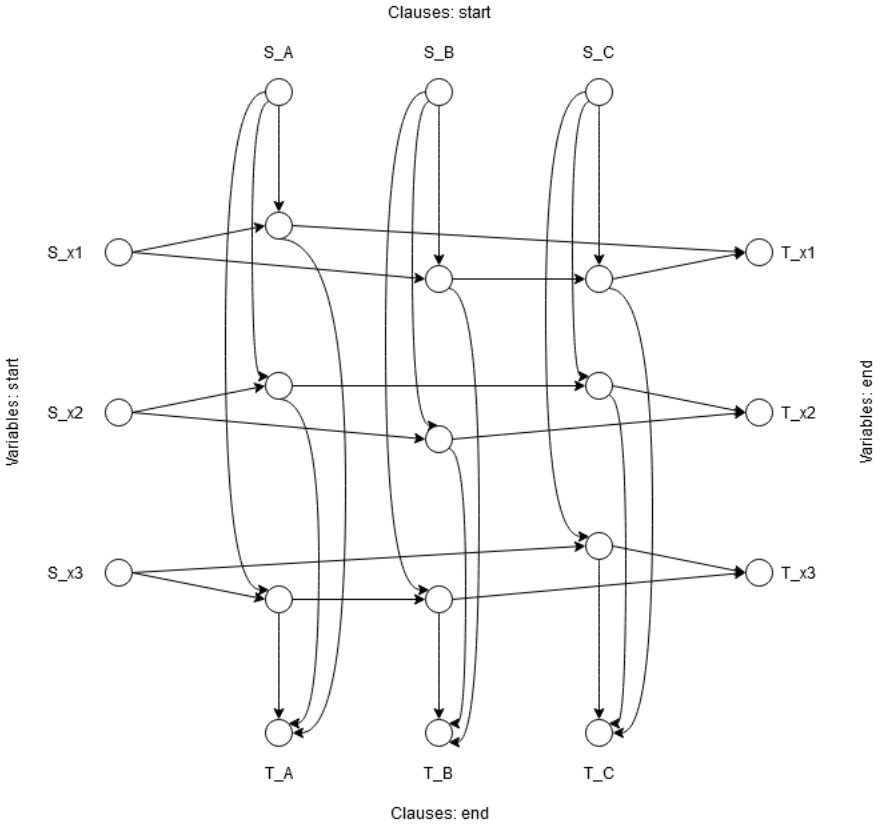
Complete this step for all variables. Note that while there are only 3 variables and 3 clauses in the example, there may be many more of each.

For each clause, there are (technically, up to) 3 variables, so there are 3 vertices in the "column" along with the s and t variables for that clause. For each of these 3 variables, add an edge from s to the vertex, and an edge from the vertex to t.



all 3 variables, first clause

Repeat this process for each clause.



final graph for $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_2 \vee x_3)$

Run the algorithm for DisjointDirectedPaths on this graph with each pair of s-t vertices as path endpoints, and return what it returns.

1.3 Conclusion

For this graph, choosing a variable to have an assignment of true would correspond to the bottom path being taken (touching vertices corresponding to clauses where the variable is negated), and an assignment of false would correspond to the top path (touching vertices corresponding to clauses where the variable is not negated) being taken.

The intuition is that for each clause (the vertical paths), it will be possible to create a path from s to t if and only if at least one of the 3 variables is not false, so in the graph at least one of the 3 vertices is not used in a horizontal path.

Thus, each individual clause will only be able to complete it's s-t path when at least one of the variables of the set satisfies the clause. Across all variables and clauses, it will only be possible to complete all paths if and only if there is a satisfying assignment for the 3SAT problem. Thus, 3SAT $\leq$ DisjointDirectedPaths, and the transformation as described runs in polytime.

Since 3SAT is a known NP-hard problem, and we have shown 3SAT is polytime reducible to DisjointDirectedPaths, DisjointDirectedPaths must also be NP-Hard.