Analysis of Algorithms I

k - Vertex Disjoin Path Problem

The given problem "The mutually avoiding path problem" or the "k - Vertex disjoint paths problem" is NP-Complete. We prove it be reducing 3-SAT to it. [1] [2]

Given a solution of k vertex disjoint paths, we simply check if each of the path's vertices and edges do exist in the main graph in polynomial time by traversing the adjacency matrix/list of the solution and comparing values with the adjacency matrix/list of the main graph. Also, to check the disjoint-ness, we can create an array of booleans of size $|V|$, where $|V|$ is as many vertices in the main graph. Each slot represents each vertex's presence in the k disjoint paths. Now, we simply perform the travesal described above again and mark each vertex in the array. If during the traversal any one slot of the array already has a truth value present in it, then the paths are not disjoint. If until the end of the traversal that does not happen then they are a valid solution. Clearly, this is a polynomial time operation as well. As verifying the solution is a polynomial time operation the problem is in NP.

Let the 3-SAT problem have m clauses and n variables. To overarching idea is to construct paths representing a satisfying assignment to each variable and to each clause and relating them in such a way that there should be $m + n$ independent paths to represent a successful assignment.

For each variable x_i $(1 <= i <= n)$ let there be a start and end vertex vs_i, vt_i. And let there be as many vT_{ij}, vF_{ij} $(1 <= j <= m)$ vertices as there are occurences of that variable in the m clauses . We connect vs_i to the first vT_{ij} and connect that to the next vT_{ij} and so on until the kast vT_{ij} and then connect that to vt_i. We repeat the same and connet $vs_i, vF_{ij}..., vt_i$ the same way. This way we have 2 paths going out from vs_i and 2 paths converging on vt_i. Each of those paths represent a truth assignment or a false assignment for that variable.

Next, for each clause c_i $(1 <= i <= m)$ let there be a start and end vertex cs_i, ct_i. And let there be as many l_{ij} vertices as there are literals in that clause $(i <= j <= 3)$. We first connect cs_i to all of the l_{ij}. Now if l_{ij} is the unnegated variable x_k, then we connect it to vF_{ki} (i.e the False vertex for variable x_k for the current clause i). If l_{ij} is the negated variable x_k, then we connect it to vT_{ki} (i.e the Truth vertex for variable x_K for the current clause i). Then we connect the vF_{ki} or the vT_{ki} chosen to ct_i. We do this for all clauses and each literal variable for that clause.

This conversion of the 3-SAT to a graph problem conveys k vertex disjoint paths problem.

There needs to be vertex independent paths for each variable and each clause. If we pick, say the truth assignment for variable x_i, then the path $vs_i \to vT_{i1} \to vT_{i2} \cdots \to vt_i$ is taken. Which means, all the clauses that have this variable x_i in the unnegated form still have vertices left to choose from to create their own paths as they were connected to the vF_{ij}. Similarly, those clauses which have this variable in the negetaed form and are thus connected to the vT_{ij} can no longer form a path as vT_{ij} vertices have already been chosen. This is representative of the clause because if x_i is True, then the clause with x_i acheives its truth value from x_i (and thus has a path using vF_{ij}), and the clause with x_i' cannot acheive its truth value from x_i (and thus cannot form a path using vT_{ij}).

This conversion of 3-SAT to the graph takes polynomial time as we add a fixed number of vertices for each clause and variable, i.e. polynomial in the size of the input.

Now, given a $k = m+n$ vertex disjoint paths solution to the converted problem, extracting the truth assignment is polynomial time operation too. Visit each vertex vs_i and check which way it branches out. If it branches out to vT_{ij}, then variable x_i has a truth value. If it branches out to vF_{ij}, then it has a false value. This is the 3-SAT satisfying assignment.

To prove that if there is no solution to the k - vertex disjoint paths problem then there is no 3-SAT solution, let us prove the contrapositive. i.e. if there is a 3-SAT assignment, then there is a k vertex disjoint paths solution. Let there be a truth assignment such that variable x_i is true. As already discussed above, in the k - vertex disjoint paths problem there will be a path for variable x_k in the form of $vs_k \to vT_{k1} \to vT_{k2} \cdots \to vt_k$. And simultaneouly that path's selection will still qualify every other clause c_i that has x_k to have a vertex disjoint path via $cs_i \to vF_{ki} \to ct_i$. This applied for each variable. And so, if a 3-SAT solution exists then the k - vertex disjoint paths problem solution also exists.

Therefore, the k - vertex disjoint paths problem is NP-Complete.

With the vertex disjoint constraint this is a very easy problem. All we have to do is apply k DFS searches starting from a_i to b_i. Each DFS operation is polynomial time complexity, so the entire k DFS operations stays polynomial too.

[3]

References

[1] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[2] Yuval Filmus (https://cs.stackexchange.com/users/683/yuval filmus). Understanding the np-completeness of induced disjoint paths. Computer Science Stack Exchange. URL:https://cs.stackexchange.com/q/110024 (version: 2019-05-29).

[3] Team. Akhil Konda, Shyam Pandya, Mausam Agrawal, Akhil Ravipati.