COMP523 Tutorial 6 - Solutions

Coordinator: Aris Filos-Ratsikas Demonstrator: Michail Theofilatos

December 27, 2019

Problem 1

Consider the *Open pit mining* problem: There is a set of blocks to be mined, each with a cost c_i and a payoff p_i and in order to mine two blocks i and i', it is required to first mine the block j directly above them. The goal is to find a set S of blocks to mine in order to maximise the profit $\sum_{i \in S}(p_i - c_i)$.

Formulate the problem as a maximum flow problem and explain how to use a solution to the maximum flow problem in order to obtain a solution to the open pit mining problem.

Solution

First, we add a source s and a sink t and a vertex i for every block to be mined. Next, for any vertex i, if $p_i - c_i > 0$, then we add a directed edge (s, i) with capacity $p_i - c_i$. Likewise, for any vertex i, if $p_i - c_i \leq 0$, then we add a directed edge (i, t) with capacity $c_i - p_i$. Finally, for every two blocks i, j, such that block i is required in order to mine block j, we add a directed edge (i, j) to the network with capacity ∞. We will prove that a minimum (s-t) cut in this network will give us the optimal set of blocks to mine, in order to maximise the profit; these will be the blocks corresponding to vertices in $S - \{s\}$. With that established, we can run a flow network algorithm on our designed network and find the minimum (s-t) cut.

Let (S, T) be an (s-t) cut in the network. For (S, T) to be minimum, there can not be an edge of infinite capacity *crossing the cut* (i.e., going from an edge of S to an edge of T or vice-versa), as otherwise the capacity would be infinity. This means that all the blocks in the set $S - \{s\}$ that we will mine will statisfy the preequisite condition, meaning that if we mine a block, we will also mine every block that is required for that block to be mined.

Now, consider the capacity of the cut (S, T). We have:

$$\begin{aligned}
c(S,T) &= \sum_{i \in T:(p_i-c_i)>0} (p_i - c_i) + \sum_{i \in S:(p_i-c_i)\leq 0} (c_i - p_i) \\
&= \sum_{i \in T:(p_i-c_i)>0} (p_i - c_i) - \sum_{i \in S:(p_i-c_i)\leq 0} (p_i - c_i) \\
&= \sum_{i \in T:(p_i-c_i)>0} (p_i - c_i) + \sum_{i \in S:(p_i-c_i)>0} (p_i - c_i) - \sum_{i \in S:(p_i-c_i)\leq 0} (p_i - c_i) - \sum_{i \in S:(p_i-c_i)>0} (p_i - c_i) \\
&= \sum_{i \in V:(p_i-c_i)>0} (c_i - p_i) - \sum_{i \in S} (p_i - c_i)
\end{aligned}$$

Looking at the right-hand side of the last equation, we observe that the first sum does not depend on the cut (S, T) and is therefore a constant. The capacity of the cut is minimised when the quantity $\sum_{i \in S}(p_i - c_i)$ is maximised, and this is precisely the mining profit. Therefore, the maximum miniming profit is achieved at the minimum cut.

Problem 2

Recall that a k-colouring of a graph G is a function $f : V \rightarrow \{1, 2, \ldots, k\}$ mapping nodes to *colours*, such that for any nodes u and v such that $(u, v) \in E$, it holds that $f(u) \neq f(v)$.

Consider the 3-colouring problem: Given a graph G as input, decide whether there is a 3-colouring of G. Prove that 3-colouring is NP-complete.

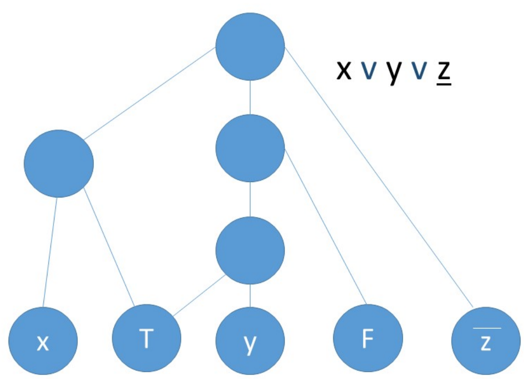
Solution

It is easy to see that 3-colouring is in NP: if we are given an assignment of colours to the vertices, we can check in polynomial time whether there exist neighbouring vertices with the same colour. To how the NP-hardness of the problem, we will construct a polynomial time reduction from 3SAT. Let ϕ be a 3SAT formula. We will have the following gadgets.

Gadget 1: A triangle of three nodes, labelled T, F and O. The colour assigned to T will be interpreted as "true" for ϕ and the colour assigned to F will be interpreted as "false". The colour assigned to O will simply be the third colour.

Gadget 2: For each variable v in ϕ, construct two vertices v and $\bar{v}$. Add an edge $(v, \bar{v})$ and edges (v, O) and $(\bar{v}, O)$ (i.e., $v, \bar{v}$ and O form a triangle).

Gagdet 3: This is the more complicated gadget shown in the figure (corresponding to the clause $x \vee y \vee \bar{z}$).



Suppose that ϕ is satisfiable, and let x be a satisfying assignment. For a variable v, if v is set to true, colour the corresponding vertex by T (and since $\bar{v}$ is set to false, colour $\bar{v}$ by T). Likewise, if v is set to false, colour the corresponding vertex v by F and $\bar{v}$ by T. Node that since v and $\bar{v}$ are connected only to O and $\bar{v}$ and v respectively in the Gadget 2 triangles, that part of the graph is 3-colourable. It remains to assign colours to Gadget 3, avoiding having any neighbours with the same colour.

Looking at the gadget of the figure, label the non-labelled vertices $1, 2, 3, 4$ starting from the left and then starting from the top in the middle column. Consider vertex 1, which is a neighbour of both x and T. If x is coloured T, then vertex 1 can be set to either O or F and if x is coloured F, then it must be set to O. Similarly for vertex 4, which is a neighbour of both T and y. If $\bar{z}$ is coloured T, we can set vertex 2 to F, vertex 1 to O, vertex 2 to T and vertex 4 to O and we have a 3-colouring. If $\bar{z}$ is set to F, then we consider the labelling of vertices x and y. Considering a few cases, we can verify that there is always a 3-colouring of the gadget.

Supppose now that we have a 3-colouring of the graph. Then, for every vertex u is coloured either T or F, we set the corresponding variable in ϕ to true or false accordingly. From the fact that the labelling is a

3-colouring and the way the graph is constructed, we know that two nodes v and $\bar{v}$ cannot receive the same colour, and therefore it is not possible for both variable v and its negation to receive the same value in the truth assignment. Finally, the correctness of the reduction follows from the fact that it is not possible for any vertex v or $\bar{v}$ to receive the colour O, because all of these vertices are connected to a vertex coloured O, and that would violate the fact that the graph is 3-colourable.