CS577: Homework 9

Haruki Yamaguchi
hy@cs.wisc.edu

Keith Funkhouser
wfunkhouser@cs.wisc.edu

November 19th, 2015

1 Algorithm description

Input: $\alpha_1, \alpha_2, \ldots, \alpha_n > 0$ and $\omega_1, \omega_2, \ldots, \omega_n > 0$, where α_i represents the cost of purchasing component i through Alpha and ω_i the cost of purchasing it through Omega. Also, the incompatibility costs $c(i, j)$ for each $i, j \in [1, n], i \neq j$.

Output: A purchasing strategy $\gamma_1, \gamma_2, \ldots, \gamma_n$, where each γ_i is A or Ω depending on which company component i should be purchased from so as to minimize the sum of the purchase costs and incompatibility costs.

Initialize the graph to have n vertices, labelled $1, 2, \ldots, n$. Henceforth we will refer to this set of vertices as N, where $|N| = n$. Also create two vertices s and t, the source and sink of our network, respectively. For each $i \in N$, create two edges:

1. An edge (s, i) with capacity ω_i
2. An edge (i, t) with capacity α_i

For each pair of vertices $u, v \in N, u \neq v$, create an edge (u, v) with capacity $c(u, v)$ corresponding exactly to the given incompatibility cost associated with components u and v being purchased from different suppliers.

Finally, run the max-flow algorithm on this network. Afterward, process the residual network to determine which vertices reside in S and which in T. All $V_\omega \in S \backslash s$ correspond to components which should be purchased from Alpha, and those $V_\alpha \in T \backslash t$ correspond to components which should be purchased from Omega. Return the purchasing strategy $\gamma_1, \gamma_2, \ldots, \gamma_n$ where each γ_i is A if vertex i is in S or Ω if vertex i is in T.

2 Correctness

The key insight here is that we want to partition the components into those that we buy from Alpha and those that we buy from Omega, and a cut is a natural way to express this. We now aim to argue that an $S - T$ cut produced by the output of the max-flow corresponds to a purchasing strategy for the components. Indeed, there is a one-to-one and onto correspondence between $S - T$ cuts and purchasing strategies for components. Furthermore, the minimal $S - T$ cut corresponds to exactly the minimum total sum of purchase and incompatibility costs.

Cuts $\rightarrow$ purchasing strategies: For a given cut, every vertex $v \in N \setminus s, t$ is either on the S or the T side. This partitioning is exactly the purchasing strategy that we are looking for, namely to purchase each component from either of Alpha or Omega.

Purchasing strategies $\rightarrow$ cuts: For a given purchasing strategy (i.e. each component will either be purchased from Alpha or Omega, there is exactly one $S - T$ cut corresponding to it, which is just the cut where those purchased from Alpha are on the S side of the cut, and those purchased from Omega on the T side.

The two correspondences are also inverses. Namely, if we start with an $S - T$ cut of G, determine the corresponding purchasing strategy, and then generate an $S - T$ cut from that strategy, we end up with

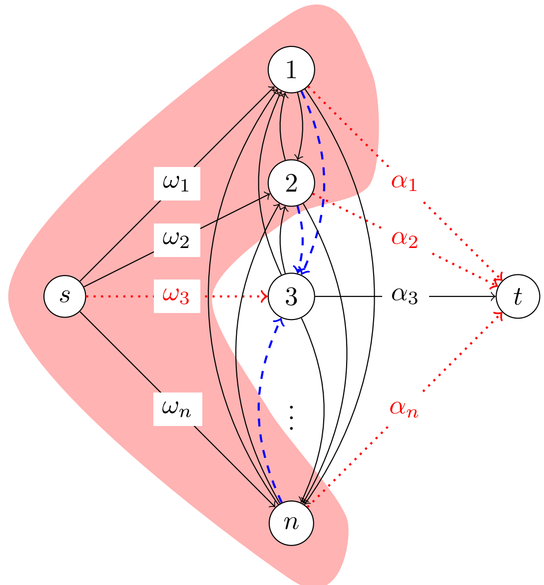

Figure 1: $S - T$ cuts correspond to purchasing strategies, and the capacity of the cut is precisely the cost of purchasing components in $S \setminus s$ from Alpha and in $T \setminus t$ from Omega. Here, components 1, 2, and n will be purchased from Alpha and incur costs of α_1, α_2, and α_n (dotted, right). Component 3 will be purchased from Omega and incurs a cost of ω_3 (dotted, left). Furthermore, the incompatibility costs are accounted for by the dashed edges. For example, the edge $(n, 3)$ has capacity $c(n, 3)$, capturing the incompatibility costs described in the problem.

identically the same cut we started with. Similarly, given a purchasing strategy we can generate an $S - T$ cut, and going from that cut back to a purchasing strategy will generate an identical strategy to the original.

Furthermore, by the max-flow min-cut theorem, the maximum value of any flow in a network is equal to the minimum capacity of any $S - T$ cut in the network [1]. Since each $S - T$ cut in G corresponds to a purchasing strategy, and the capacity of said cut is exactly the objective function of cost, the $S - T$ cut produced by the max-flow will determine an optimal purchasing strategy to minimize costs.

3 Runtime

The constructed graph is composed of $n^2 + 2n = O(n^2)$ edges and $O(n)$ vertices. Furthermore, it is constructed in $O(n^2)$ time, because construction of the $2n$ source and sink edges $\omega_1, \omega_2, \ldots, \omega_n$ and $\alpha_1, \alpha_2, \ldots, \alpha_n$ can be done in $O(n)$ time and the n^2 incompatibility cost edges $c(i, j)$ can be constructed in $O(n^2)$ time. The state-of-the-art max-flow algorithm discussed in class runs in $O(|V| \cdot |E|)$. Because of the structure of this graph, this is $O(n^2 \cdot n) = O(n^3)$.

Once we have run the max-flow algorithm, we just find a minimum cut in the flow network, and those vertices on the S side of the cut tell us which components to buy from Alpha (and those in T tell use which to buy from Omega).

References

[1] Dieter van Melkebeek. Network flow, October 2015.