1 Overview

Instead of maximizing $(\sum \text{ANCH})^3 + (\sum \text{ANSH})^3$, we maximized $M(\sum \text{ANCH}) + (\sum \text{ANSH})$ for a very huge constant M. If we ignore the triple/twin constraints, the problem can be solved by using a minimum-cost flow algorithm. Using this relaxation lower bound, we solved the original problem by the branch-and-bound method.

2 Details

2.1 Problem

The problem can be formulated as follows. Given a bipartite graph $G = (U \cup V, E)$, an edge-weight function $w : E \to \mathbb{R}$, a multiplicity function $d : U \to \mathbb{R}_{>0}$, and a capacity function $c : V \to \mathbb{R}_{>0}$, find a minimum-weight subset $M \subseteq E$ satisfying (i) $|M \cap \delta(u)| = 1$ and (ii) $\sum_{uv \in M \cap \delta(v)} d(u) = c(v)$.

2.2 Relaxation

The LP relaxation of this problem is the minimum-cost flow problem, which can be solved in polynomial time. More particularly, we construct a network as follows. Each edge $uv \in E$ has capacity $d(u)$ and cost $w(uv)/d(u)$. Each vertex $u \in U$ has supply $d(u)$, and each vertex $v \in V$ has demand $c(v)$. In the original problem, we have additional constraints that, for every edge $uv \in E$, its flow value must be either 0 or $d(u)$.

2.3 Branch and Bound

For solving the original problem, we use a branch-and-bound method. For every node of the search tree, we do as follows.

First, we compute a relaxed solution by solving the minimum-cost flow problem (we implemented a cost-scaling algorithm by Goldberg and Tarjan). At the root node, we need to solve the relaxed problem from scratch, but for the other nodes, we can obtain a relaxed solution in $O(|E| \log |V|)$ time by modifying the relaxed solution for the parent node by searching an augmenting path.

If the obtained relaxed solution is larger than the solution for the original problem we have found so far, we quit the subsequent search and backtrack to the parent node. Otherwise, we choose an edge uv whose flow value $f(uv)$ is neither 0 nor $d(u)$. If there are no such edges, we obtain an improved solution for the original problem and backtrack to the parent node. Otherwise, we branch into two cases: $f(uv) = 0$ or $f(uv) = d(u)$. In order to make the search space smaller, we compute a relaxed solution for every unsatisfying edge and choose the one with the largest lower bound.

2.4 Reductions

Because the input is very large, we cannot solve it directly. First, we remove all the edges of weight zero and relax the constraints (i) and (ii) to (i') $|M \cap \delta(u)| \leq 1$ and (ii') $\sum_{uv \in M \cap \delta(v)} d(u) \leq c(v)$. From the solution for this relaxed problem, we construct a solution for the original problem by solving the knapsack problem.

Second, we apply the reduced-cost fixing as follows. Let D be the difference between a known upper bound and the current lower bound. Let f be the minimum-cost flow and let $p : U \cup V \to \mathbb{R}$ be the dual variables (i.e., potentials).

For edge $uv \in E$ with $f(uv) < d(uv)$, the increase of the cost for fixing $f(uv) = d(u)$ is at least $(w(uv) + p(u) - p(v))(d(u) - f(uv))$. Therefore, if this value is larger than D, we can fix $f(uv) = 0$.

For edge $uv \in E$ with $f(uv) > 0$, the increase of the cost for fixing $f(uv) = 0$ is at least $f(uv)\ell(u, v)$, where $\ell(u, v)$ is the shortest-path distance from u to v in the residual graph. Therefore, if this value is larger than D, we can fix $f(uv) = d(u)$. Because $|V|$ is small (1000), we can compute all the distances $\ell(u, v)$ by using the Dijkstra's algorithm against the reversed graph.