Tsinghua University

Discrete Optimistic

Fall 2024

Homework 1

Lin Zejin April 13, 2025

- **Collaborators:** I finish this homework by myself.

Problem 1. (a) Since if flow f with value v is feasible in this network, then λf with value λv, $\lambda > 1$ is feasible in the network. So if the min-flow exists, there exists a feasible flow with large enough value.

Choose $C = \sum_{e \in E} l(e)$. Then C can be a possible value of a feasible flow. Find a feasible flow in this circulation network, where the demand of source and sink is C, by using max-flow algorithm.

Then we find a feasible flow f in this network.

Let $c(e) = f(e) - l(e), e \in E$. Then for each feasible flow g, $g(e) \geq l(e) \Rightarrow f(e) - g(e) \leq c(e)$. And the flow conversation remains:

$$\sum_{e=(x,v)} (f(e) - g(e)) - \sum_{e=(v,x)} (f(e) - g(e)) = 0$$

Then to find a min-flow g, suffices to find a max-flow $f - g$ with capacity $c(e)$.

It is a polynoimial algorithm in $O(|V|, |E|, \log C)$.

(b) There is no analogue for this problem. Note that if there exists a circulation, then the min-flow value is 0. However, max-cut value cannot reach 0 even if we change the definition.

Problem 2. For each maximum flow f and min-cut (A, B), since

$$\text{val}(f) = \sum_{e:A\to B} f(e) - \sum_{e:B\to A} f(e) = \sum_{e:A\to B} c(e) = c(A, B)$$

we have $f(e) = 0$ for every e from nodes in B to nodes in A and $f(e) = c(e)$ for every e from nodes in A to nodes in B. Actually, it is also the sufficient condition for min-cut (A, B).

Thus, use the Capacity-Scaling Algotihm to find a maximum flow f and min-cut (A, B) in polynomial time, and check each $a \in A$ to see whether it is central or upstream and each $b \in B$ to see whether it is central or downstream.

For $a \in A$, if there is a min-cut (A', B') such that $a \in B'$. Then if x can be reached from a with all edges of positive weight, or with all non-satuarted edges, $\Rightarrow x \in B'$, otherwise $\exists e$ from B' to A' with $f(e) > 0$, or $\exists e$ from A' to B' with $f(e) \neq c(e)$ respectively.

So denote $M = B \cup \{x : x$ can be reached from a with all edges of positive weight, or with all non-satuarted edges$\}$, $N = V \setminus M$. Note that each edge from N to M is saturated and each edge from M to N is zero. Thus (N, M) is a min-cut if $N \neq \emptyset$. However, if $N = \emptyset$, it means $a \in M$, which causes contradiction since $a \in B'$ as we discuss above.

In short, for each $a \in A$, we only need to construct M_a in polynomial time and check whether $M_a = A$, if so, a is upstream, otherwise, a is central.

Similar for $b \in B$. It needs time complexity $O(nm)$.

So the total time complexity only depends on the time complexity of algorithm to find a single max-flow.

Problem 3. As we have set an algorithm to classify each node to be central, upstream or downstream. If there is some central nods, it can't be the unique min-cut. If there is no central nodes, it means all nodes are upstream or downstream. So the unique min-cut is (A, B) where A is the set of all upstream nodes and B is the set of all downstream nodes.

Problem 4. (a) Construct a bipartite graph G with two visual nodes s, t.

s points to all ballons, with capacity 2 and all ballons point to conditions they can measure with capacity 1, and conditions point to t with capacity k.

Since max-flow in integer network has integer value, it suffices to check whether the max-flow is saturated for edges to t, if so, the edges of weight 1 in G is what balloons exactly measure, and if not, there is no possible solution.

(b) We can add the pair of condition and subcontractor, denoted as (s_i, c_j).

s points to all ballons, with capacity 2 and all ballons point to the pair of conditions they can measure and corrosponding subcontractors with capacity 1, and each pair (s_i, c_j) point to c_j with capacity 1, and conditions point c_j to t with capacity k.

Then any subcontractor can only measure each condition once, (note that it is equivalent to the orignial problem since multi-measure only causes waste) and each condition will be measured k times if the edge is saturated.

Since max-flow in integer network has integer value, it suffices to check whether the max-flow is saturated for edges to t, if so, the edges of weight 1 from balloons to pairs is what balloons exactly measure, and if not, there is no possible solution.

Problem 5. (a) We construct $T + 3$ visual nodes like below, where T is the number of edges in M:

First each node y covered by M points to a visual node n_y with capacity 1.

Second, each node x uncovered by M points to the same visuaol node n, with capacity 1.

Third, each node x in X is pointed by the source s, wtih capacity 1, and n_y, n all points to the sink t, with capacity 1 (for n_y) and k (for n) respectively.

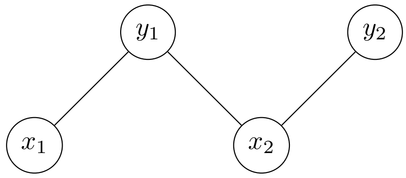
Edges in the original graph is equipped with capacity ∞.

Then a feasible flow with integer value should be a matching as before. If edges to t are all saturated, then nodes covered by M are covered by the new matching, with k uncovered nodes in addition, which induces a possible solution.

Note that a possible solution have to correspond a max-flow in the network. So it suffices to check whether the max-flow is saturated for edges to t, if so, the chosen edges of weight 1 from X to Y are exactly edges of M', and if not, there is no possible solution.

The running time complexity is just the same as the time complexity of algorithm to find a single max-flow with $n + T + 3$ nodes and $m + n + T + 1$ edges, which is $O(n(m + 2n)^2)$

(b) The bipartite $\{x_1, x_2\}$ and $\{y_1, y_2\}$ like below:



If $M = \{(y_1, x_2)\}$, then the coverage expansion can only be $\{(x_1, y_1), (x_2, y_2)\}$ which does not intersect with M.

(c) It suffices to prove the network in (a) and the network in the lecture to find maximum matching returns the same value of min-cut.

For any min-cut (A, B), since the edges from X to Y have capacity ∞, there is no edges from A to B that belongs to the origninal biparite graph. So $c(A, B)$ in (a) only calculates two parts: capacity from the source s to $X \cap B$ and the capacity in Y side.

Since $t \in B$, whether the middle nodes n_y belongs to A or B, it only calculates the capacity 1 from $y \in A$ to n_y once. And if $n \in A$, then capacity k from n to t is calculated. Because the corresponding max-flow implies there is no edges from y uncovered by M' to n cross A and B, so $k = |A \cap \{y \text{ uncovered by } M'\}|$. And if $n \in B$, it will calculates all capacity 1 from $A \cap \{y$ uncovered by $M'\}$ to n once.

In short, $c(A, B) = |X \cap B| + |Y \cap A|$ for any k.

However, to find maximum matching, the min-cut capacity is also $|X \cap B| + |Y \cap A|$.

Then the size of maximum matching is $K_2 \Rightarrow \exists(A, B)$, $|X \cap B| + |Y \cap A| = K_2 \Rightarrow k = K_2 - |M'|$ is feasible for coverage expansion $\Rightarrow K_1 \geq k + |M'| \geq K_2$.

So $|K_1| = |K_2|$

Problem 6. We can obtain a directed graph G in polynomial time that describes the nesting relation ship between each pair of boxes. Now suffices to find a minimum number of disjoint paths in G that covers all nodes in G.

Now we use it to construct a network.

First, there is a source node s, pointing to all boxes i_{in}, and a sink node t, pointed by all boxes i_{out}, with capacity ∞. Here i_{in}, i_{out} have the same meaning for the boxes but refers to different nodes. *i.e.* There are two nodes representing the same box.

If there is a directed edge $a \to b$ in G, which means that b is nest inside a, then we add an edge from a_{in} to the node $(a \to b)$ and an edge from the node $(a \to b)$ to b_{out}. The capacity of these edges is 1.

We say an edge in G is *chosen* if the corrosponding connected edge in the network is saturated.

For this network, a feasible flow with integer value should satisfy:

1.1. edges in G like $a \to x$ for x unknown will be chosen once since the capacity to a_{in} is 1.

1.2. edges in G like $x \to b$ for x unknown will be chosen once since the capacity from b is 1.

So each nodes in G connects with the input edge and the output edge that is chosen once.

Therefore, the chosen edges in G forms several disjoint paths and some discrete nodes.

Note that, the value of flow will be $\sum_{(a\to b)\in G} f(a_{in} \to (a \to b))$ which is the number of the chosen edges, denoted as V. Since several disjoint paths and some discrete nodes are actrually a forest, V also equals to $n-$the number of trees in the forest. So the number of trees in the forest is $n - V$, which implies that to minimize the number of trees in the forest, we actuall are to maximize the value of flow.

So algorithm will be like this:

1.1. Construct a directed graph G in polynomial time that describes the nesting relation ship between each pair of boxes.

1.2. Construct a network with G as we discussed above.

1.3. Use the Capacity-Scaling Algorithm to find a maximum flow in the network.

1.4. The number of disjoint paths is $n - V$, where V is the value of flow.

$n - V$ is what we need.

The running time complexity is just the same as the time complexity of algorithm to find a single max-flow with at most $2 + 2n + \frac{n(n-1)}{2}$ nodes and $n(n-1) + 2n$ edges, which is $O(n^5)$.

Problem 7. Let $L = \infty$. First, we have a network combined with all subnetwork for each pixel $i \in V$ of the same virtual s, t.

For any $k = 1, 2, \cdots, M$, neighborhood pair (i, j), we set the edge $(v_{i,k}, v_{j,k})$ and $(v_{j,k}, v_{i,k})$ with capacity p_{ij}.

If for min-cut of the network, there are two lowcapacity edges a_{i,k_1}, a_{i,k_2}. Assume $k_1 < k_2$. Then the path from v_{i,k_2} to v_{i,k_1+1} crosses A to B so there exists a highcapacity edge that leaves A

Then for any min-cut of the graph, it contains exactly one lowcapacity edge a_{i,d_i} that leaves A for each i, and moreover, if neighborhood (i, j) satisfies $d_i > d_j$, then all of edges from $v_{i,t}$ to $v_{j,t}$, $d_j + 1 \le t \le d_i$ will be calculated in the calculation of min-cut value.

Therefore, the value will be

$$\sum_{k=0}^{M} \sum_{d_i=k} a_{i,k} + \sum_{k<l} \sum_{(i,j)\in E, d_i=k, d_j=l} (l-k)p_{ij}$$

So each min-cut corresponds to a labeling $A_k = \{i : d_i = k\}$.

Since for each labeling we can construct a corresponding cut with the same value, the min-cut value is what we need. So minimum-cost labeling can be efficiently computed from a minimum s-t cut in this network.

Problem 8. If $(u, v), (v, u) \in E$, it is a linear problem for minimizing $a(u, v)f(u, v) + a(v, u)f(v, u)$, which can be reduced to the case of one edge. For simplicity, we assume that if $e \in E$, then $e_{reverse} \notin E$.

(a) If there is a circle C with negative cost in the residual network of f, let $f'(e) = \begin{cases} f(e) & e, e_{reverse} \notin C \\ f(e) + \delta & e \in C \cap E \\ f(e) - \delta & e_{reverse} \in C \end{cases}$ where

$\delta = \min\{f(e) : e \in C\}$. Then

$$\sum_{e\in E} a(e)f'(e) = \sum_{e\in E} a(e)f(e) + \sum_{e\in C\cap E} a(e)\delta + \sum_{e\in C\setminus E} a(e)(-\delta) = \sum_{e\in E} a(e)f(e) + \delta \sum_{e\in C} a(e) < \sum_{e\in E} a(e)f(e)$$

So f cannot be the minimum cost flow.

If there is no circle C with negative cost, *i.e.* each circle has postitive cost in the residual network.

Assume g is the min-cost flow, $g \neq f$.

Consider the flow t obtained by $g - f$.

For $e \in E$, $t(e) = g(e) - f(e)$ if $g(e) - f(e) > 0$, otherwise $t(e_{reverse}) = -(g(e) - f(e))$.

Noticed that the cost remains the same:

$$\sum_{e} a(e)t(e) = \sum_{e\in E} a(e)(g(e) - f(e)) + \sum_{e\notin E}(-a(e))(f(e) - g(e)) = \sum_{e\in E} a(e)(g(e) - f(e))$$

If there is a circle with positive cost sum, let $g'(e) = \begin{cases} g(e) & e, e_{reverse} \notin C \\ g(e) - \delta & e \in C \cap E \\ g(e) + \delta & e_{reverse} \in C \end{cases}$ with δ small enough. Then g' has less cost, which causes contradiction!

(It is better to set a lemma that the cost remains for the transition)

Since all edges in t positive, but noticed that $\sum_{(s,u)} t(s) - \sum_{(u,s)} t(s) = 0$, it is circular. So there has to be some circle in the flow $\Rightarrow$ there is a circle of non-zero value with negative cost sum.

So this circle can't be in the graph $G_f \Rightarrow \exists e$ in the circle, e has 0 capacity in graph G_f, which impies $f(e) = c(e)$.

If $e \in E$, then $g(e) \geq f(e) = c(e) \Rightarrow g(e) = c(e) \Rightarrow t(e) = 0$ contradiction!

If $e \notin E$, then $g(e) \leq f(e) = c(e) = 0 \Rightarrow t(e) = 0$ contradiction!

So f has to be the min-cost flow.

(b) We prove that each f in the loop satisfies G_f has no circles with negative cost sum.

If not, $\exists$ status f *s.t.* G_f has no circles with negative cost sum but after one iterations it has.

Assume after one iteration, f becomes g and G_g has a negative circle C. P_f is the path with minimum cost in G_f

Then C doesn't belong to G_f.

However, since the direction of C is only affected by P_f, P_f intersects with C if we ignore the direction.

P_f changes some edges $e_1, e_2, \cdots, e_n$ of C, but not whole of C. So there is another path if we remove $e_1, \cdots, e_n$ and add other edges of C. Its cost is less than P_f since the sum of C is negative, which means exactly

$\sum_{i=1}^{n} c(e_i) \geq \sum_{e\in C\setminus\{e_1,\cdots,e_n\}} c(e)$.

So this is the contradiction. Therefore it gives a min-cost flow.

(c) Algorithm in (b) actually returns a feasible max-flow which has no circle with negative cost sum in its residual network. Thus it is also a min-cost flow. The time compleixty is $O(|v| \cdot |E| \cdot C) \cdot O(|V|^2)$ when we use the Dijstra algorithm to find the min-cost path.

(d) By the result of P5, max-cost matching is also the matching with maximum edges. (Otherwise, it can be expanded to a maximum matching, whose cost will not be less than it)

So it suffices to find the min-cost max-flow in the network constructed in the lecture.

Explicitly, s points to nodes in A with capacity 1 and cost 0. Edges in E have capacity of ∞ and its cost and t is pointed by nodes in B with capacity 1 and cost 0.

The answer equals to the result.