Algorithms Design and Analysis
Homework Assignment 3

Dumb Ways to Die

November 9, 2015

3 Flows, Matchings, and Linear Programming

Terminologies and Symbols

In this paper, all undirected edges are represented as $\{u, v\}$ with $u \neq v$ which is a set, and all directed edges are represented as (u, v) which is an ordered pair.

3.1 Graph Orientations With Degree Constraints

Exercise 3.1.

1. We can construct a network $G_f = (V_f, E_f, c_f)$ such that

$$V_f = \{s, t\} \cup \{e_{\{u,v\}} | \{u, v\} \in E\} \cup \{v_k | k \in V\}$$

and

$$E_f = \{(s, e_{\{i,j\}}) | \{i, j\} \in E\} \cup \{(e_{\{i,j\}}, v_i), (e_{\{i,j\}}, v_j) | \{i, j\} \in E\} \cup \{(v_k, t) | k \in V\}$$

with capacity function $c_f : E_f \mapsto \mathbb{R}^+$

$$c_f(e') = \begin{cases} 1 & e' = (s, e_{\{i,j\}}) \text{ for } \{i, j\} \in E \\ 1 & e' = (e_{\{i,j\}}, v_i) \text{ or } (e_{\{i,j\}}, v_j) \text{ for } \{i, j\} \in E \\ c(k) & e' = (v_k, t) \text{ for } k \in V \end{cases}$$

If there is a maxflow of value $|E|$, we know that there is a integral flow of value $|E|$.

For any $\{i, j\} \in E$, it has $f((s, e_{\{i,j\}})) = 1$.

And for any edge, we have $\{i, j\} \in E$, $f((e_{\{i,j\}}, v_i)) = 1 \wedge f((e_{\{i,j\}}, v_j)) = 0$ or $f((e_{\{i,j\}}, v_i)) = 0 \wedge f((e_{\{i,j\}}, v_j)) = 1$.

So we can orientate edge $\{i, j\}$ to (j, i) when $f((e\{i, j\}, v_i)) = 1$ or to (i, j) when $f((e\{i, j\}, v_j)) = 1$.

Now every edge in E has exactly one orientation. Note that $f((v_k, t)) \leq c(k)$, so $deg^{in}_H(k) \leq c(k)$. Thus we get a feasible orientation of G with respect to the degree constraints c.

Or the maxflow's size is less than $|E|$. Obviously there is no feasible orientation of G where every edge should have exactly one orientation.

But there may be some feasible orientations where some edges has two orientations.

Consider there is a feasible orientation of G where some edges has two orientations. Then we can remove one of the orientations of these edges and we can get a feasible orientation. There comes a contradiction. So if the maxflow's size is less than $|E|$, feasible orientation of G with respect to the degree constraints c does not exist.

Now consider there is a feasible orientation of G. If there are some edges with two orientations, we can remove one of the orientation of these edges and get a new feasible orientation where every edge has exactly one orientation. We can set the flow function as follows.

(a) Because every edge has exactly one orientation, for any $\{i, j\} \in E$, we have $f((s, e_{\{i,j\}})) = 1$(Similarly, $f((e_{\{i,j\}}, s)) = -1$).

(b) $f((e_{\{i,j\}}, v_i)) = 1$ for $\{i, j\} \in E$ oriented to (j, i)(Similarly, $f((v_i, e_{\{i,j\}})) = -1$).

Now the flow is conserved for all $e_{\{i,j\}}$. Then let $f((v_k, t)) = deg^{in}_H(k)$ for $k \in V$(Similarly, $f((t, v_k)) = -deg^{in}_H(k)$). Thus, flow is conserved for all v_k. Obviously, $f \leq c$. Then we get a flow of value $|E|$. And obviously, the cut $(\{s\}, V\backslash\{s\})$ tells us that the sizes of flows in G_f can not be greater than $|E|$.

So f is a maxflow of G_f.

Now we need to calculate the complexity of this reduction. There are $2 + |V| + |E|$ vertexes in G_f and $|V| + 3|E|$ edges in G_f. We see that $|V_f| = O(|V| + |E|) = O(|V| + |V|^2) = O(|V|^2)$ and $|E_f| = O(|V| + 3|E|) = O(|V|^2)$ are both of polynomial size with respect to $|V|$. And since the maxflow problem can be solved in polynomial time, the whole algorithm costs only polynomial time with respect to $|V|$.

2. Construct a bipartite graph $G_b = (V_b, E_b)$ with its vertices' set $V_b = \{v_{i,k} | i \in V, k = 1, 2, \ldots, c(i)\} \cup \{e_{\{u,v\}} | \{u, v\} \in E\}$, every $v_{i,k}$ standing for the vertex i in the original graph, every $e_{\{u,v\}}$ standing for the edge $\{u, v\}$ in the original graph.

   The k from 1 to $c(i)$ here means that we split vertex i into $c(i)$ identical vertices.

   The edges in the bipartite graph are

   $$E_b = \{(v_{u,k}, e_{\{u,v\}}) | \{u, v\} \in E, k = 1, 2, \ldots, c(i)\}$$

   If we have gotten a complete matching $M \subseteq E_b$ in graph G_b, then we orientate edge $\{u, v\}$ to (u, v) iff $(v_{v,k}, e_{\{u,v\}}) \in M$ for some k.

   Note: "Complete" here means $|M| = |E|$. That is every vertex in the bipartite graph corresponding to the edges in the original graph has an orientation.

Or if we have found a feasible solution for the orientation problem, then we sort the edges $E_v = \{(u,v)|\{u,v\}\ is\ oriented\ to\ (u,v)\}$ with the index number of u and put $(v_{v,k}, e_{\{u,v\}})$ in the match M iff (u,v) is the kth element in E_v.

Note: We can not deal with the situation that some edges have been oriented to both direction. But if we have found such a feasible solution, we can make them point to only one side and this would be another feasible solution.

Now we need to calculate the complexity of this reduction. Since we split a vertex into at most $\max\{c(i)\}$ parts. then the size of vertices' set $|V_b|$ of the new bipartite graph is at most $O(|V|\max\{c(i)\}+|E|)$ and the size of edges' set is at most $O(|E|\max\{c(i)\})$.

Since $c(i)$ can be represented in digits, our reduction might not be polynomial. But we can prove that if we make $c(i) = \min\{c(i), |E|\}$, the answer would not change.

The proof is simple. We can see that $c(i)$ means the maximum degree that vertex i is restricted to have. But in any feasible solution, the degree would never exceed the size of edges' set $|E|$. Then if $c(i) > |E|$, the restriction is meaningless, we can make it to $|E|$ without the change of the answer.

Exercise 3.2.

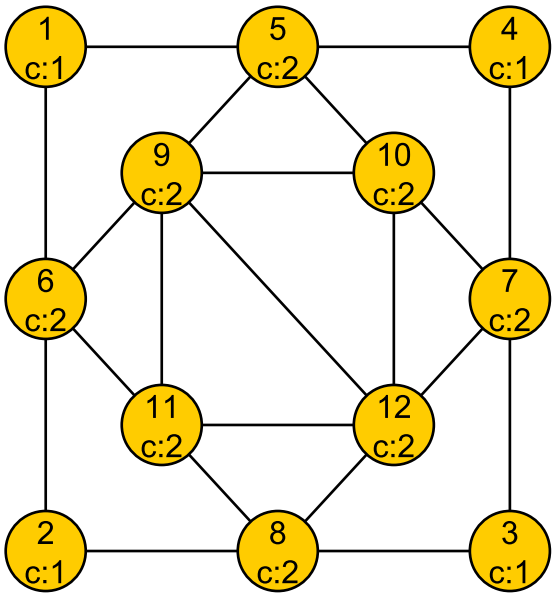


Figure 1: A graph without feasible orientation.

1. There is no feasible orientation in Figure 1, because

$$\sum_{\forall v \in V} c(v) = 20 < 21 = |E|$$

2. Firstly, define "related vertex" set $r(Y)$ of a set of edges $Y \subseteq E$.

$$r(Y) := \{u, v | \exists \{u, v\} \in Y\}$$

Then G has no feasible orientation with respect to c if and only if there is a set $Y \subseteq E$ with $\sum_{x \in r(Y)} c(x) < |Y|$.

3. *Proof.* From **Exercise 3.1.1** above we know that an orientation problem can be reduced to a maxflow problem. Consider thers is a set $Y \subseteq E$ with $\sum_{x \in r(Y)} c(x) < |Y|$.

Define $A \colon = \{e_{\{u,v\}} | \{u, v\} \in Y\}$, $B \colon = \{v_k | \exists e_{\{i,j\}} \in A \wedge c((e_{\{i,j\}}, v_k)) > 0\}$, $U_1 \colon = \{e_{\{u,v\}}\}$ and $U_2 \colon = \{v_k\}$.

We can construct a s-t cut $(X, V_f \backslash X)$ with

$$X = \{s\} \cup A \cup B$$

Accodring to the definition of B, no edge in E_f will comes from A and goes to $U_2 \backslash B$. So

$$c(X, V_f \backslash X) = \sum_{e_{\{u,v\}} \in U_1 \backslash A} 1 + \sum_{v_k \in B} c(k)$$

Accodring to the definition of B

$$\begin{aligned} B \colon \quad &= \{v_k | \exists e_{\{i,j\}} \in A \wedge c((e_{\{i,j\}}, v_k)) > 0\} \\ &= \{v_i, v_j | \exists e_{\{i,j\}} \in A\} \\ &= \{v_i, v_j | \exists \{i, j\} \in Y\} \end{aligned}$$

Compare it with the definition of $r(Y)$

$$r(Y) := \{u, v | \exists \{u, v\} \in Y\}$$

Obviously there is a bijection b from $r(Y)$ to B:

$$\begin{array}{rccc} b \colon & r(Y) & \longrightarrow & B \\ & x & \longmapsto & v_x \end{array}$$

Thus

$$\sum_{x \in r(Y)} c(x) = \sum_{v_k \in B} c(k)$$

Finally

$$\begin{aligned} c(X, V_f \backslash X) &= \textstyle\sum_{v \in A} 1 + \sum_{v_k \in B} c(k) \\ &= \textstyle\sum_{v \in A} 1 + \sum_{x \in r(Y)} c(x) \\ &< \textstyle\sum_{v \in A} 1 + |Y| \\ &= |U_1| \\ &= |E| \end{aligned}$$

Here, the cut $c(X, V_f \backslash X)$ is a "witness" for the non-existence of a flow of value $|E|$. So G has no feasible orientation with respect to c. □

Exercise 3.3.

Note: The variables i, j below stands for all team except team 1.

We can construct a graph $G = (V, E)$ such that

$$V = \{s, t\} \cup \{match_{\{i,j\}}\} \cup \{team_i\}$$

and

$$E = \{(s, match_{\{i,j\}})\} \cup \{(match_{\{i,j\}}, team_i)\} \cup \{(team_i, t)\}$$

with capacity function $c : E \mapsto \mathbb{R}^+$

$$c(e) = \begin{cases} m_{i,j} & e = (s, match_{\{i,j\}}) \\ +\infty & e = (match_{\{i,j\}}, team_i) \\ \hat{s}_1 - s_i - 1 & e = (team_i, t) \end{cases}$$

where $\hat{s}_1 = s_1 + \sum_{i \neq 1} m_{1,i}$ which is the highest score that team 1 can get.

Now we translate the flow problem into the game's language.

The vertex standing for $match_{\{i,j\}}$ receive flows from s, with the amount up to $m_{i,j}$, then it should "distribute" these flows This means that every match must have an unique winner and the winner team $team_i$ or $team_j$ receives the flows from $match_{\{i,j\}}$.

But, team 1 wants to be the unique winner. As a result, the score $\hat{s}_i$ that team i finally get should strictly less than the score $\hat{s}_1$ that team 1 finally get. Since scores are integers

$$\hat{s}_i = s_i + f(team_i) \leq \hat{s}_1 - 1$$

That means

$$f(team_i) \leq \hat{s}_1 - s_i - 1 \tag{1}$$

It is exactly the capacity constrain.

Then we observe that the flow network is an integral network, it has at least one maximum solution with integer flow F.

For the flow F, $F(match_{\{i,j\}}, team_i)$ equals to the number of the matches that team i wins with team j.

If the value of the flow satisfies that

$$val(F) = \sum_{\{i,j\}} m_{i,j}$$

which means F can "distribute" all the matches' result to its participant with team 1's "winner constrain" (1). Then it is possible (if all goes well) that team 1 can become the unique winner.

For the complexity analysis, we see that $|V| = O(n + n^2) = O(n^2)$ and $|E| = O(3n^2 + n) = O(n^2)$ is both of polynomial size with respect to n. And since the maximum flow problem can be solved in polynomial time, then the whole algorithm costs only polynomial time with respect to n.

3.2 Linear Programming

Exercise 3.4.

1. Give a feasible solution achieving a value of at least 30.

   Sol. We can see
   $$x = 6, y = 3, z = 3$$
   which satisfies
   $$\begin{aligned} 6 - 3 &= 3 \leq 3 \\ 6 + 3 - 3 &= 6 \leq 6 \\ 2 \cdot 3 - 3 &= 3 \leq 6 \\ 2 \cdot 3 - 6 &= 0 \leq 3 \end{aligned}$$
   is a feasible solution.
   $$3x + 3y + 3z = 3 \cdot 6 + 3 \cdot 3 + 3 \cdot 3 = 36$$

2. Derive an upper bound on the optimum by multiplying and adding the constrains. Your bound should be at most 100.

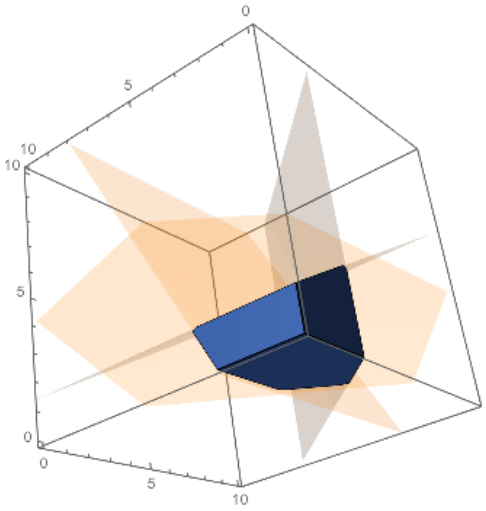
   Sol.
   $$\begin{aligned} x - y &\leq 3 && (1) \\ x + y - z &\leq 6 && (2) \\ 2y - z &\leq 6 && (3) \\ 2z - x &\leq 3 && (4) \end{aligned}$$
   Multiply and add the constrains,
   $$\begin{aligned} & 3 \cdot (1) + 4 \cdot (2) + 1 \cdot (3) + 4 \cdot (4) \\ \Rightarrow & (3x - 3y) + (4x + 4y - 4z) + (2y - z) + (8z - 4x) \\ & \leq 3 \cdot 3 + 4 \cdot 6 + 1 \cdot 6 + 4 \cdot 3 = 51 \end{aligned}$$
   Thus 51 is an upper bound on the optimum.

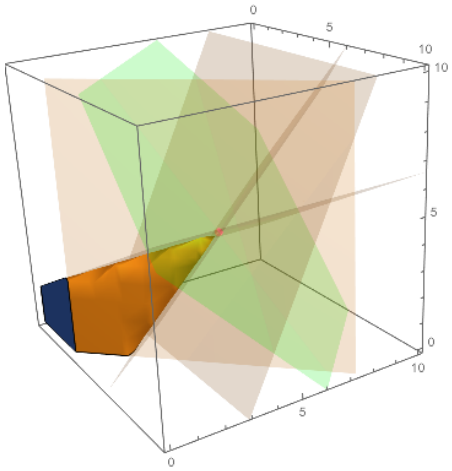
3. Find the optimal solution P.

   Sol. The optimal solution is
   $$x = 7, y = 4, z = 5$$
   in this case $3x + 3y + 3z = 48$ is the maximum.

4. Prove your solution is optimal by multiplying and adding the constrains appropriately.

   Proof. Multiply and add the constrains as
   $$\begin{aligned} & 2 \cdot (1) + 5 \cdot (2) + 4 \cdot (4) \\ \Rightarrow & (2x - 2y) + (5x + 5y - 5z) + (8z - 4x) \\ & \leq 2 \cdot 3 + 5 \cdot 6 + 4 \cdot 3 = 48 \end{aligned}$$
   Thus 48 is an upper bound on the optimum and we achieve it by evaluating $x = 7, y = 4, z = 5$. Hence 48 is exactly the optimal solution. □





5. Construct P^*, the dual of P.

Sol. P^*:

$$\begin{aligned} \text{minimize} \quad & 3a + 6b + 6c + 3d \\ \text{subject to} \quad & a + b - d \geq 3 \\ & -a + b - 2c \geq 3 \\ & -b - c + 2d \geq 3 \\ & a, b, c, d \geq 0 \end{aligned}$$

is the dual of P.