CSE530 : Algorithm and Complexity Assignment II

JongYun Kim (20185053)

PROBLEM I

We are going to have two steps to find a minimum $s-t$ cut in the given graph.

(i) *Find maximum flow for the given network.*

(ii) *Find a cut which has the maximum value of a flow.*

By the *Max-flow min-cut theorem*, the cut from the step (ii) becomes a minimum cut in the given graph.

(i) ***Find maximum flow for the given network.***

STEP I) Find a shortest path

As shown in Figure 1, we found a shortest path, namely p_1. The residual capacity is $c_f(p_1) = 1$.

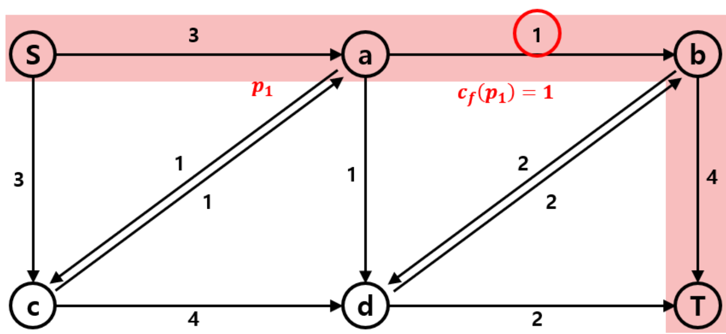


Fig. 1. shortest path p_1

STEP II) Augmenting the path p_1 with $c_f(p_1)$ and find a shortest path again

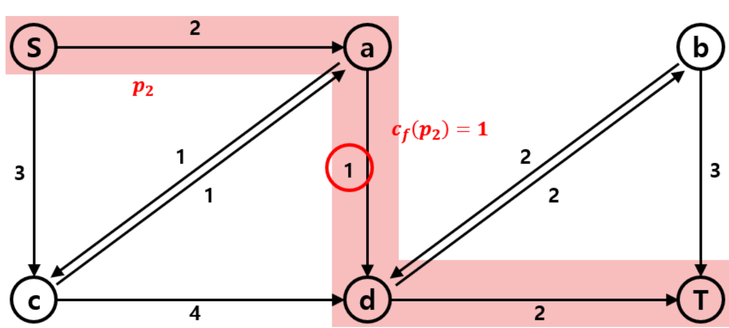


Fig. 2. shortest path p_2

We can push $c_f(p_1) = 1$ units through p_1 at the Figure 1. Then Figure 2 is given. We can get the next shortest path p_2 and $c_f(p_2) = 1$.

STEP III) Augmenting the path p_2 with $c_f(p_2)$ and find a shortest path again

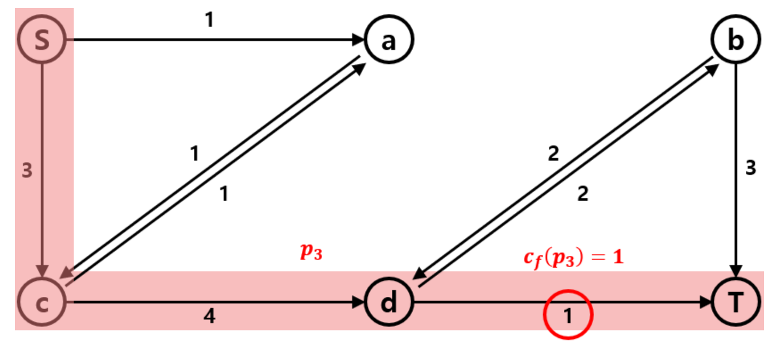


Fig. 3. shortest path p_3

We can push $c_f(p_2) = 1$ units through p_2 at the Figure 2. Then Figure 3 is given. We can get the next shortest path p_3 and $c_f(p_3) = 1$.

STEP IV) Augmenting the path p_3 with $c_f(p_3)$ and find a shortest path again

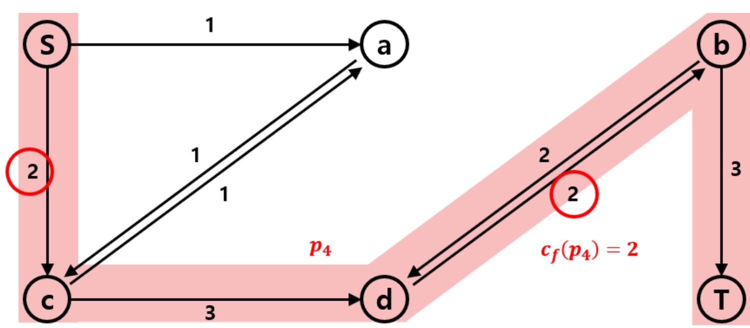


Fig. 4. shortest path p_4

We can push $c_f(p_3) = 1$ units through p_3 at the Figure 3. Then Figure 4 is given. We can get the next shortest path p_4 and $c_f(p_4) = 2$.

STEP V) Augmenting the path p_3 with $c_f(p_3)$ and check whether you can find a new path or not

We can push $c_f(p_3) = 1$ units through p_3 at the Figure 4. Then Figure 5 is given. We, however, cannot find a new path in this time. The sink t is unreachable. So the algorithm(The Edmonds-Karp algorithm) terminates. Not that the total augmented flow value is $|f|_t = 1 + 1 + 1 + 2 = 5$.

Note: vertex **T** should have been denoted by t in Figure 1-5.

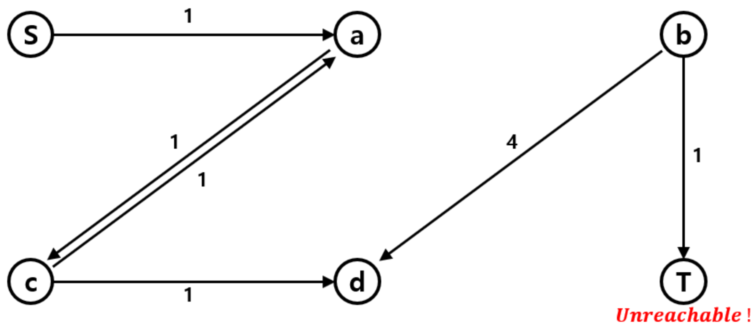


Fig. 5. after augmenting the path p_4

(ii) ***Find a cut which has the maximum value of a flow.***

As previously mentioned, $|f|_t = 1 + 1 + 1 + 2 = 5$. So we have to find a cut which has $c(S, T) = |f|_t = 5$.

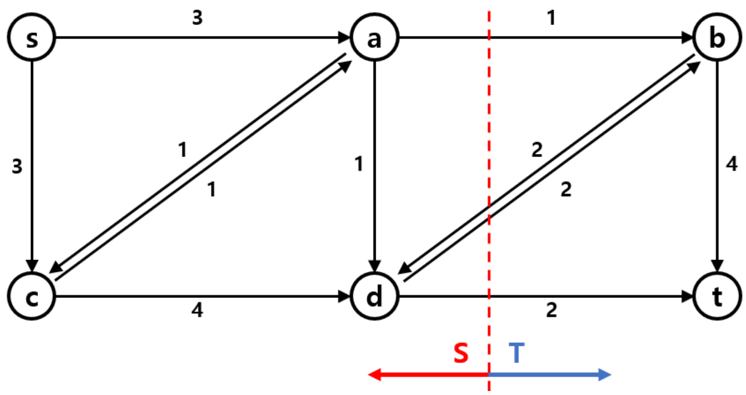


Fig. 6. The minimum $s - t$ cut in the given graph

$$\begin{cases} S = \{s, a, c, d\} \\ T = \{b, t\} \end{cases}$$

PROBLEM II

Problem II-(a)

Let x_1, x_2, and x_3 be the number of Radio A, B, and C respectively. And consider the below conditions.

(i) The objective is the revenue over one week.

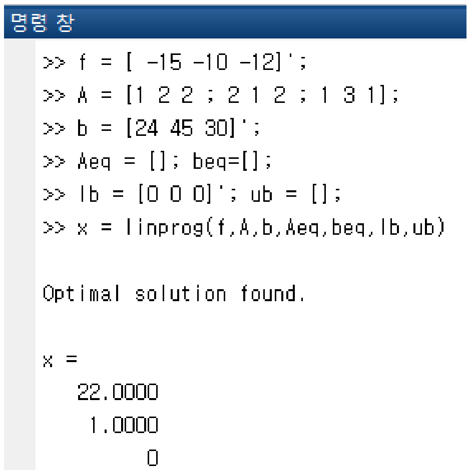
(ii) The three people have each limit of work time over one week.

(iii) The number of Radio produced should be non-negative.

$$\text{maximize} 15x_1 + 10x_2 + 12x_3$$

$$
\begin{aligned}
\text{subject to} \quad x_1 + 2x_2 + 2x_3 &\leq 24 \\
2x_1 + 1x_2 + 2x_3 &\leq 45 \\
x_1 + 3x_2 + x_3 &\leq 30 \\
x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

Problem II-(b)



```
명령 창
>> f = [ -15 -10 -12]';
>> A = [1 2 2 ; 2 1 2 ; 1 3 1];
>> b = [24 45 30]';
>> Aeq = []; beq=[];
>> lb = [0 0 0]'; ub = [];
>> x = linprog(f,A,b,Aeq,beq,lb,ub)

Optimal solution found.

x =
   22.0000
    1.0000
         0
```

Fig. 7. The linear program is solved by a linear program solver in MATLAB

The LP is solved by the linear program solver in MATLAB as shown in Figure 7. The solution is given below.

$$15x_1 + 10x_2 + 12x_3 = 340$$

, where

$$
\begin{cases}
x_1 = 22 \\
x_2 = 1 \\
x_3 = 0
\end{cases}
$$

PROBLEM III

Problem III-(a)

Suppose that

$$
\begin{aligned}
x_1 &= x_3 - x_4 \\
x_2 &= x_5 - x_6 \\
x_3, x_4, x_5, x_6 &\geq 0
\end{aligned}
$$

And we may want to consider one more thing as follows.

$$|x_1| + |x_2| \leq 1 \qquad \Leftrightarrow \qquad \begin{cases} x_1 + x_2 & \leq 1 \\ x_1 - x_2 & \leq 1 \\ -x_1 + x_2 & \leq 1 \\ -x_1 - x_2 & \leq 1 \end{cases}$$

Now, we are able to change the form of the problem as follows.

$$\begin{aligned} \text{maximize} \quad & x_3 - x_4 + 2x_5 - 2x_6 \\ \text{subject to} \quad & x_3 - x_4 + x_5 - x_6 \quad \leq 1 \\ & x_3 - x_4 - x_5 + x_6 \quad \leq 1 \\ & -x_3 + x_4 + x_5 - x_6 \quad \leq 1 \\ & -x_3 + x_4 - x_5 + x_6 \quad \leq 1 \\ & x_3, x_4, x_5, x_6 \quad \geq 0 \end{aligned}$$

This is the linear program in standard form. Since any linear program can be written in a standard form, the problem given in PROBLEM III is a linear program.

Problem III-(b)

We are going to use Simplex algorithm. We first convert the form in the previous sub-problem into a slack form.

$$\begin{aligned} z &= x_3 - x_4 + 2x_5 - 2x_6 \\ x_7 &= 1 - x_3 + x_4 - x_5 + x_6 \\ x_8 &= 1 - x_3 + x_4 + x_5 - x_6 \\ x_9 &= 1 + x_3 - x_4 - x_5 + x_6 \\ x_{10} &= 1 + x_3 - x_4 + x_5 - x_6 \\ & x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \geq 0 \end{aligned}$$

The basic solution corresponding to the slack form is

$$(x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (0, 0, 0, 0, 1, 1, 1, 1) \qquad with \quad z = 0$$

Increasing x_3 would increase z. But x_3 is limited to increasing by 1 because of the equations of x_7 and x_8. The equation $x_7 = 1 - x_3 + x_4 - x_5 + x_6$ can be rewritten into the following form.

$$x_3 = 1 + x_4 - x_5 + x_6 - x_7$$

If we substitute this equation for the problem (*i.e.* let x_3 and x_7 be entering and leaving variable), we get

$$\begin{aligned} z &= 1 + x_5 - x_6 - x_7 \\ x_3 &= 1 + x_4 - x_5 + x_6 - x_7 \\ x_8 &= 2x_5 - 2x_6 + x_7 \\ x_9 &= 2 - 2x_5 + 2x_6 - x_7 \\ x_{10} &= 2 - x_7 \\ & x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \geq 0 \end{aligned}$$

We can still increase z if x_5 increase. So x_5 becomes a entering variable and x_9 becomes a leaving variable.

$$\begin{aligned}
z &= 2 - \tfrac{3}{2}x_7 - \tfrac{1}{2}x_9 \\
x_3 &= x_4 - \tfrac{1}{2}x_7 + \tfrac{1}{2}x_9 \\
x_5 &= 1 + x_6 - \tfrac{1}{2}x_7 - \tfrac{1}{2}x_9 \\
x_8 &= 2 - x_9 \\
x_{10} &= x - x_7 \\
& x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \geq 0
\end{aligned}$$

The basic solution of this slack form is given as follows.

$$x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (0, 0, 1, 0, 0, 2, 0, 2) \qquad with \quad z = 2 \tag{1}$$

We cannot increase $z = 2 - \frac{3}{2}x_7 - \frac{1}{2}x_9$ anymore while increasing the non basic variables. So we've reached the optimal point of this problem. Therefore, we get the solution.

$$max(x_1 + 2x_2) = 2 \qquad , where \quad \begin{cases} x_1 = 0 \\ x_2 = 1 \end{cases}$$

PROBLEM IV

If *the vertex capacity constraint* can be reduced into a standard maximum flow problem, we are able to use *the Edmonds-Karp algorithm* which has $O(|V| \cdot |E|^2)$ running time.

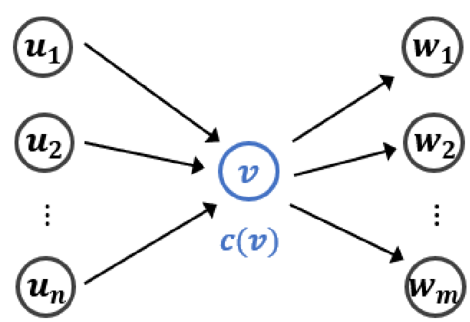


Fig. 8. Generalized maximum flow problem with capacity constraint

We can think a vertex v with the given vertex capacity $c(v)$ in a given graph $G(V, E)$. The vertex v has n inlet vertexes $u_1, \cdots, u_n$ and m outlet vertexes $w_1, \cdots, w_m$ as shown in Figure 8. Suppose that we split the vertex v into two vertexes v_i and v_j, where $c(v_i, v_j) = c(v)$. Then we can get a new network $G(V^*, E^*)$ as shown in Figure 9. It is worth noting that the network becomes $G(2V - 2, E + V - 2)$. Because all vertexes in the original graph

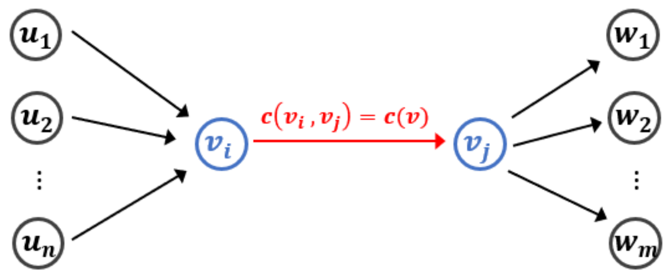


Fig. 9. Generalized maximum flow problem with capacity constraint has been reduced into the standard maximum flow problem with edge capacity $c(v_i, v_j = c(v)$

$G(V, E)$ except source s and sink t are split into two vertexes each. So we've got additional $(V + 2)$ vertexes and edges.

$$\begin{aligned} V^* &= V + (V - 2) = 2V - 2 \\ E^* &= E + (V - 2) \end{aligned}$$

In the network $G(V^*, E^*)$, the vertex constraint is reduced as holding the other 3 constraints. So it is obvious that $G(V, E)$ and $G(V^*, E^*)$ are equivalent and the new network $G(V^*, E^*)$ belongs to a standard maximum maximum flow problem. If we use *the Edmonds-Karp algorithm*, it takes $O(|V^*| \cdot |E^*|^2)$ running time.

$$|E| \geq |V| - 1 \tag{2}$$

$$2|E| \geq |E| + |V| - 1 \geq |E| + |V| - 2 \tag{3}$$

Every connected graph follows Equation (2). And mathematically Equation (2) and (3) are equivalent. Then we can get the following things.

$$\begin{aligned} |V^*| \cdot |E^*|^2 &= |2V - 2| \cdot |E + V - 2|^2 \\ &\leq |2V| \cdot |2E|^2 \\ &\leq 8|V| \cdot |E|^2 \end{aligned} \tag{4}$$

Equation (4) directly implies

$$O(|V^*| \cdot |E^*|^2) = O(8|V| \cdot |E|^2) = O(|V| \cdot |E|^2)$$

PROBLEM V

Suppose that a rectilinear polygon P is given. We can partition the polygon P into unit square and let the partition become a part of a *chessboard*. Note that the set of unit squares is V. Let's say black squares belong to L and white squares belong to R. It is obvious that $V = L + R$ and $L \cap R = \emptyset$. Let E be Edge where two vertexes are connected to each other if a vertex is adjacent to the other. Now we can construct bipartite graph G(V, E)

TABLE I: Partitioning a rectilinear polygon with rectangles

Algorithm 2
procedure PARTITION(polygon P)
$V \leftarrow$ unit square partition of polygon P
$S \rightarrow$ the set of the sum of x and y coordinate of each unit square
for $i \leftarrow 1$, $\|V\|$-1 **do**
if $S[i]$ is odd number **then**
$L \leftarrow V[i]$
else
$R \leftarrow V[i]$
for $i \leftarrow 1, \|V\| - 1$ **do**
for $j \leftarrow i + 1, \|V\| - 1$ **do**
if $V[i]$ is adjacent to $V[j]$ **then**
$E[i][j], E[j][i] \leftarrow (V[i], V[j]), (V[j], V[i])$
$C[i][j], C[j][i] \leftarrow 1$
Create the graph G(V, E)
$V' \leftarrow V + \{s, t\}$
$E' \leftarrow E + \{(s, u); u \in L\} + \{(v, t); v \in R\}$
$\|f^*\|, matching \leftarrow Edmonds - Karp\ algorithm(G'(V', E'))$
$M^* \leftarrow \|f^*\|$
if $M^* = \frac{\|V\|}{2}$ **then**
return $matching$
else
return *False*

from the polygon P. If we are able to get the maximum bipartite matching M^*, we can verify the polygon can be partitioned into 2×1 and 1×2 rectangles by checking $M^* = (\text{the number of unit squares})/2$. Because finding the maximum matching of graph G(V, E) is equivalent to covering the polygon P with 2×1 and 1×2 rectangles (*i.e.* partitioning P into the rectangles).

$$M^* = \begin{cases} \frac{|V|}{2} & \rightarrow P \text{ can be partitioned} \\ otherwise & \rightarrow P \text{ can not be partitioned} \end{cases} \tag{5}$$

The value M^* in Equation (5) can be computed by finding the maximum flow of the *corresponding flow network* G'(E', V') for the bipartite graph G, where

$$\begin{aligned} V' &= \{V \cup \{s, t\}\} \\ E' &= \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R, and (u, v) \in E\} \cup \{(v, t) : v \in R\} \end{aligned}$$

Note that s is source and t is sink of the flow network G'. Once we've got the flow network G', we can use *Edmonds-Karp algorithm* which takes $O(|V'||E'|^2)$.

The pseudo code is given in Table I. Let n be the area of polygon P. $n =$ area of P $= \frac{|V|}{2}$. Most of the line runs within $O(n^2)$ except line 17 where Edmonds-Karp algorithm is running.

$$\begin{aligned} |V| &= 2n \\ |E| &\leq 4n^2 \end{aligned}$$

The first equation is derived from the definition of n. The inequality is also true because every vertex has at most 4 connections, as considering the polygon P and partition into unit square S.

$$|V'| = |V| + |\{s,t\}| = 2n + 2 \tag{6}$$

$$|E'| = |V| + |E| + |V| \leq 4n^2 + 2n \tag{7}$$

$$O(|V'||E'|^2) = O((2n+2)(4n^2+2n)^2) = O(n^5) \tag{8}$$

Therefore, we can conclude that the algorithm in Table I has $O(n^5)$ running time.