Algo HW5 산경 김관호

1. objective: $\min\left(\max_i |ax_i + by_i - c| \right)$

Let's introduce objective variable $z \geq 0$

Obj: minimize z s.t.

$$\begin{cases} |ax_1 + by_1 - c| \leq z \\ |ax_2 + by_2 - c| \leq z \\ \vdots \\ |ax_n + by_n - c| \leq z \end{cases}$$

Resolve to linear problem by reduce absolute values.

Obj: min z

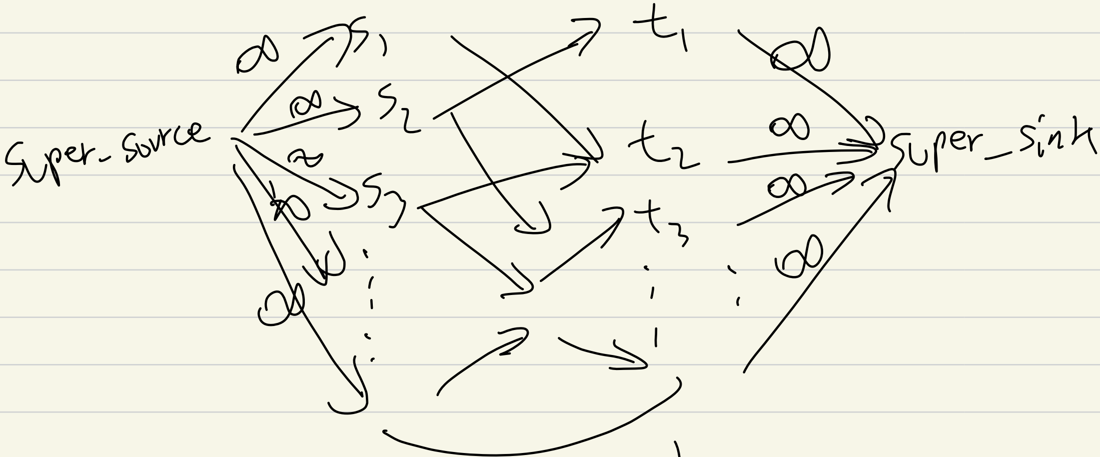
$$\left.\begin{aligned} ax_1 + by_1 - c &\leq z \\ -ax_1 - by_1 + c &\leq z \\ ax_2 + by_2 - c &\leq z \\ -ax_2 - by_2 + c &\leq z \\ &\vdots \\ ax_n + by_n - c &\leq z \\ -ax_n - by_n + c &\leq z \end{aligned}\right\} \text{constraints}$$

where $z \geq 0$ and (x_i, y_i) is given point in problem $i = 1, \cdots, n$

By above way, we can find z that is minimum value of maximum absolute error.

2. (a)

Introduce super_source which give flows all sources $s \in \{s_1, s_2, \dots\}$ and super-sink which sinks all sinks:

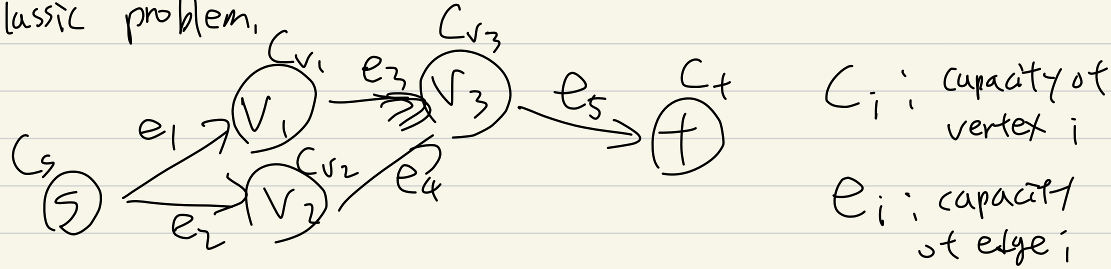


Original Graph

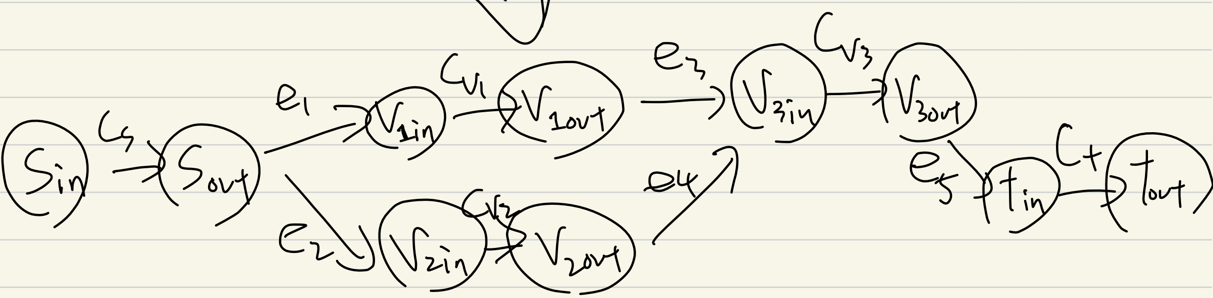
As above drawing, connect super_source to all sources and super-sink to all sinks.
And set capacity each super_source to s_i and t_i to super-sink INF.
In this way, we can modify multi source & sink problem into single source & sink max-flow problem.

2(b)

In case each vertex has capacity, we split vertex into V_{in}, V_{out} and put directed edge which connects (V_in, v_out).
And edge's capacity is equal to V's capacity meanwhile V_in and v_out have no capacity as classic problem.



Reduce



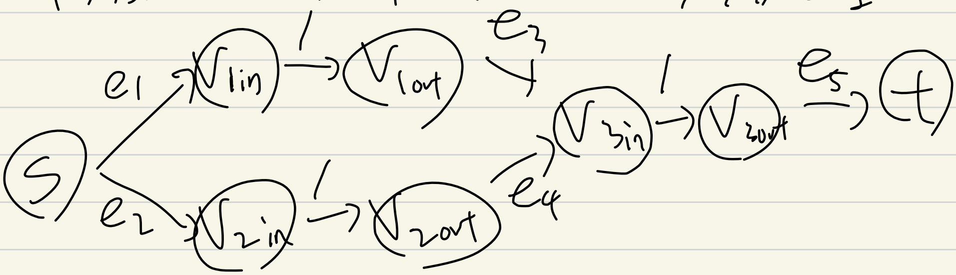
This is formal max-flow problem with single source S_{in} and single sink t_{out}.

3.

Let's combine 2(a) and 2(b) to solve this problem.

Split each vertex V into V_{in}, V_{out} and put edge (V_{in}, V_{out}) with capacity 1. It allows only one path available through that vertex.

* Assume capacity of edge $(e_1, e_2, \dots) \geq 1$



Now find max-flow. That is exactly # of vertex-disjoint paths from S to t.

Since capacity of vertex V is 1, when find max-flow of modified graph, the path which goes through one vertex is must unique.

Therefore, any path from S to t in the max-flow problem in modified graph corresponds to an element in set of vertex-disjoint paths from S to t.

4.

(a) To see whether it's NP or not, we need to check it's determined in polynomial time if any complete assignment is given.

Let's assume a potential solution (X_1, X_2, X_3, X_4) is given. Then we can check whether disjunction outcome is true or not by putting each given variable into each clause. It can be completed in polynomial time because each clause has at most four literals, time complexity is proportional to # of clauses.

Thus, it belongs to NP.

(b) To see whether it's NP-hard or not, we need to check whether a known NP-hard problem would be reduced to this problem in polynomial time or not.

Let's pick 3SAT as a known NP-hard problem and reduce it to 4SAT (= this problem)

It's simply reduced just put an extra boolean variable X_e to each clause with 'or' operation, and X_e' which is complement.

For example, when 3-SAT disjointed connection like below,

$$(X_1 \vee X_2 \vee X_3) \wedge (\bar{X}_1 \vee X_2 \vee X_3) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee X_3)$$

put X_e to every clause.

$$(X_1 \vee X_2 \cup X_3 \vee X_e) \wedge (\bar{X}_1 \vee X_2 \cup X_3 \vee X_e) \wedge (\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3 \vee X_e)$$

$$\wedge (X_1 \vee X_2 \cup X_3 \vee X_e') \wedge (\bar{X}_1 \vee X_2 \cup X_3 \vee X_e') \wedge (\bar{X}_1 \vee \bar{X}_2 \vee \bar{X}_3 \vee X_e')$$

+ explanation

If converted 4-SAT problem is true, then 3-SAT problem is also true because there are x_e, x_e'. True assignment is rather than value of x_e, x_e'.

If it's false, then for same reason, 3-SAT assignment problem is also false,

in poly-time,

5. Let's reduce well known np-complete partition problem to rectangle packing problem (let's say this problem as RP) to show RP is np-hard.

At first, modify input of partition problem.

For each element 'e' in set S, modify to rectangle whose width is |e| and height is '1'.

Then, put this modified input to RP problem adding one condition,

condition; height of containing smallest rectangle must be '2'.

If width of the smallest rectangle is more than $\frac{1}{2}$(sum of elements in set S), than there is no answer to partition problem.

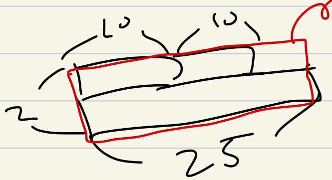
else if width of that is equal to $\frac{1}{2}$(sum of sets),

then by dividing upper and lower part of rectangle, partition problem's answer is derived.

In this manner we can reduce partition problem (np-complete) to RP problem in poly-time.

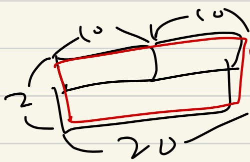
So, RP problem is np-hard

Ex)



smallest rectangle width $= 25 > \frac{1}{2}(10+10+25)$

$\Rightarrow$ there is no answer



smalles rectangle width $= 20 = \frac{1}{2}(10+10+20)$

$\Rightarrow$ ther is an answer