Algorithm HW#1

B02901178 江誠敏

March 25, 2017

1 Problem 1.

In the following, v_i and w_i will represent the value and the weight of the i-th gift respectively, W be the weight limit, and let $V \triangleq \sum v_i$. Also, let V^* be the total value of optimal solution. Assume that $w_i \leq W$ for all i (or else the gift is useless), so $V^* \geq \max v_i$.

First we summarize some result we learned in lecture 1 and HW 0.5.

Lemma 1.

1. *There is an algorithm that solves the knapsack problem in $\mathcal{O}(nV^*)$, where V^* is the total value of the optimal solution.*
2. *Fixing a problem instance, if we choose b to be our rounding factor, then the time complexity is $\mathcal{O}(nV^*/b)$, and we get an $(1 + 2nb/V^*)$-approximation algorithm.*
3. *There is an $\mathcal{O}(n \log n)$ greedy 2-approximation algorithm for the knapsack problem. Simply sort the gifts by v_i/w_i, so assume $v_i/w_i \geq v_{i+1}/w_{i+1}$, then choose $\max(v_1 + \cdots + v_k, v_{k+1})$, where k is the smallest indices letting $w_1 + \cdots + w_{k+1} \geq W$.*

So if we can choose $b = \epsilon V^*/(2n)$, then we would get an $(1 + \epsilon)$-approximation algorithm which runs in $\mathcal{O}(n^2\epsilon^{-1})$. Let $\tilde{V}$ be the solution given by the greedy algorithm in 3., then $V^*/2 \leq \tilde{V} \leq V^*$. So if we let $b = \epsilon\tilde{V}/(4n)$, it would still be an $(1 + \epsilon)$-approximation algorithm, and the running time is $\mathcal{O}(n^2\epsilon^{-1})$.

Notice that the problem could also be solved without the greedy 2-approximation algorithm. First we need a lemma.

Lemma 2. *Let $\bar{V}$ be the optimal solution after rounding, and let $\hat{V}$ be the corresponding solution in the original problem. If $\bar{V} \leq (1 + \epsilon)\hat{V}$ then $\hat{V}$ is an $(1 + \epsilon)$ approximation algorithm.*

Proof. This is because $V^* \leq \bar{V} \leq (1+\epsilon)\hat{V}$, so $\hat{V}$ is an $(1+\epsilon)$ approximation algorithm. □

So we propose an algorithm as following:

Algorithm 1: An $\mathcal{O}(n^2\epsilon^{-1})$ FPTAS for the knapsack problem

$V \triangleq \sum v_i$

$b \leftarrow \epsilon V/(2n)$

while *true* **do**

  $\hat{V} \leftarrow$ an approximation solution using the algorithm in lemma 1-2 with rounding factor b. `/* Cost` $\mathcal{O}(nV^*/b)$ `*/`

  Check if $\hat{V}$ gives a solution good enough by lemma 2. `/* Cost` $\mathcal{O}(n)$ `*/`

  if $\hat{V}$ *is good enough* **then return** $\hat{V}$

  else $b \leftarrow b/2$

end

Notice that $V^* \le V = \sum v_i$, so initially $b \ge \epsilon V^*/(2n)$. And once $b \le \epsilon V^*/(2n)$, $\hat{V}$ is then an $(1+\epsilon)$ approximation solution and the algorithm stops. At this moment, $b \ge \epsilon V^*/(4n)$. Also, $V^* \ge \max v_i \ge V/n$, so the iteration runs at most $\log n$ times. Hence the total running time is

$$\mathcal{O}(\log n) + 4n^2\epsilon^{-1} + 2n^2\epsilon^{-1} + n^2\epsilon^{-1} + \cdots = \mathcal{O}(\log n) + \mathcal{O}(n^2\epsilon^{-1}) = \mathcal{O}(n^2\epsilon^{-1})$$

Collaborators:

- B02901085 徐瑞陽: Reminds me that there is already a 2-approximation algorithm mentioned in class.

2 Problem 2.

We consider a slightly different problem: Each gift g_i has two value (u_i, v_i), and could be assigned to child A, or to child B, or to none of them. Let S_A be the gifts assigned to child A, and S_B be the gifts assigned to child B, and $S_A, S_B \ne \varnothing$ should be satisfied. Define $\mathcal{U} \triangleq \sum_{i \in S_A} u_i$, $\mathcal{V} \triangleq \sum_{i \in S_B} v_i$, The goal is to minimize $\max(\mathcal{U}, \mathcal{V})/\min(\mathcal{U}, \mathcal{V})$. We use $(\mathcal{U}, \mathcal{V})$ to denote the corresponding solution.

The original problem could be reduced to this problem simply by setting $u_i = v_i$.

Lemma 3. *If we know that the optimal solution $(\mathcal{U}^*, \mathcal{V}^*)$ satisfied $\mathcal{U}^* \le U$ and $\mathcal{V}^* \le V$, then the problem mention above could be solved in $\mathcal{O}(nUV)$.*

Proof. Consider the function $f :: (\mathbb{Z}, \mathbb{Z}, \mathbb{Z}) \to \{\texttt{true}, \texttt{false}\}$ such that $f(k, u, v)$ means that if there exist a assignment such that $\mathcal{U} = u$ and $\mathcal{V} = v$, considering only the first k

gifts. Then we could write the recursive formula as

$$f(k,u,v) = \begin{cases} \texttt{true}, & \text{if } k=0 \text{ and } u=v=0 \\ \texttt{false}, & \text{if } k=0 \text{ and one of } u,v \neq 0 \\ \texttt{false}, & \text{if } u<0 \text{ or } v<0 \\ \begin{array}{r} f(k-1,u,v) \vee f(k-1,u-u_k,v) \\ \vee f(k-1,u,v-v_k) \end{array}, & \text{otherwise} \end{cases}$$

Using memorization (i.e., dynamic programming), since we only need to consider the states (k,u,v) which satisfied $0 \le k \le n, 0 \le u \le U, 0 \le v \le V$, and each state only depends on $\mathcal{O}(1)$ states, thus a dynamic programming could calculate all possible pairs (u,v) in $\mathcal{O}(nUV)$, and we could then output $\min\limits_{f(n,u,v)=\texttt{true}} \max(u,v)/\min(u,v)$. $\square$

Notice that if we require that g_t should be assigned to a child, say child A. Then we only need to make a small modification on this recursive formula: When $k=t$, then $f(k,u,v) = f(k,u-u_t,v)$, and the other remain unchanged. The time complexity is still $\mathcal{O}(nUV)$.

Now we give a FPTAS to this problem. Since each child is assigned to at least one gift, let $\alpha = \arg\max_{i\in S_A} u_i$, $\beta = \arg\max_{i\in S_B} v_i$ for a solution. we could enumerate through possible pairs of (α,β). The $\mathcal{O}(n^2)$ possible pairs are $\{(i,j) : 1 \le i,j \le n \text{ and } i \neq j\}$.

Now, for a pair (α,β), we find an $(1+\epsilon)$-approximation optimal solution under the restriction that $\arg\max_{i\in S_A} u_i = \alpha$, $\arg\max_{i\in S_B} v_i = \beta$.

Let $b_u = \epsilon u_\alpha/(2n), b_v = \epsilon v_\beta/(2n)$. Round the value of the gifts to $\tilde{g}_i = (\tilde{u}_i, \tilde{v}_i) \triangleq (\lceil u_i/b_u \rceil, \lceil v_i/b_v \rceil)$. Let $(\tilde{\mathcal{U}}, \tilde{\mathcal{V}})$ be the optimal solution in the rounded version with the same restriction. Since rounding preserves "$\le$", i.e., $x \le y \implies \lceil x/b \rceil \le \lceil y/b \rceil$, so we know that $\tilde{\mathcal{U}} \le \sum_{i=1}^{\alpha} \tilde{u}_i \le nu_\alpha/b_u$, similarly, $\tilde{\mathcal{V}} \le nv_\beta/b_v$, then using the algorithm in lemma 3, the optimal solution could be calculate in $\mathcal{O}\left(n(2nu_\alpha/b_u)(2nv_\beta/b_v)\right) = \mathcal{O}(n^5\epsilon^{-2})$, which is polynomial in n, ϵ.

After we find the optimal solution in $\{\tilde{g}_i\}$, we take this assignment as an approximation solution of the original problem. Let this assignment to be $(\tilde{S}_A, \tilde{S}_B)$, and the optimal assignment of the original problem to be (S_A^*, S_B^*). Notice that for both assignment (call the one we're considering (S_A, S_B)),

$$\sum_{i\in S_A} b\tilde{u}_i - b \le \sum_{i\in S_A} u_i \le \sum_{i\in S_A} b_u\tilde{u}_i \implies (1-\epsilon/2)\sum_{i\in S_A} \tilde{u}_i \le \sum_{i\in S_A} u_i \le \sum_{i\in S_A} b\tilde{u}_i$$

Since we restrict that $\alpha \in S_A$, $\sum_{i\in S_A} u_i \ge u_\alpha$, so $nb = \epsilon u_\alpha/2 \le \sum_{i\in S_A} u_i$. Similarly,

$$(1-\epsilon/2)\sum_{i\in S_B} \tilde{v}_i \le \sum_{i\in S_A} v_i \le \sum_{i\in S_A} b\tilde{v}_i$$

Thus without loss of generality, assume $\mathcal{U}^* \geq \mathcal{V}^*$, then

$$\begin{aligned}
\frac{\max(\mathcal{U}^*, \mathcal{V}^*)}{\min(\mathcal{U}^*, \mathcal{V}^*)} = \frac{\mathcal{U}^*}{\mathcal{V}^*} &= \sum_{i \in S_A^*} u_i \Big/ \sum_{i \in S_B^*} v_i \\
&\geq (1 - \epsilon/2) \sum_{i \in S_A^*} \tilde{u}_i \Big/ \sum_{i \in S_B^*} \tilde{v}_i \\
&\geq (1 - \epsilon/2) \sum_{i \in \tilde{S}_A} \tilde{u}_i \Big/ \sum_{i \in \tilde{S}_B} \tilde{v}_i \\
&\geq (1 - \epsilon/2)^2 \sum_{i \in \tilde{S}_A} u_i \Big/ \sum_{i \in \tilde{S}_B} v_i \\
&\geq (1 - \epsilon) \sum_{i \in \tilde{S}_A} u_i \Big/ \sum_{i \in \tilde{S}_B} v_i
\end{aligned}$$

Hence it is an $(1 + \epsilon)$ approximation algorithm. [1]

Finally, enumerate through all possible pair (α, β), then we get an $(1 + \epsilon)$ approximation algorithm with time complexity $\mathcal{O}(n^2)\,\mathcal{O}(n^5\epsilon^{-2}) = \mathcal{O}(n^7\epsilon^{-2})$.

Collaborators:

- B02901085 徐瑞陽: Mentioned that the problem could also be solved by rounding, since the algorithm could be polynomial in $\mathcal{O}(\log W)$.

3 Problem 3.

We shall assume that $k \geq 2$, since if $k = 1$ then there is no edge and the solution is trivial.

Recall that the corresponding relaxed LP of the weighted set cover is:

$$\begin{aligned}
&\text{minimize} \quad && \boldsymbol{w}^\intercal \boldsymbol{x} \triangleq \sum_{i=1}^{m} w_i x_i \\
&\text{s.t.} && x_u + x_v \geq 1, && \forall (u, v) \in E \\
& && x_v \geq 0, && \forall v \in V
\end{aligned} \tag{1}$$

For convenience we shall assume $V = \{1, 2, \ldots, n\}$.

First we site a result proved in class:

Lemma 4. *The relaxed LP corresponded to the weighted vertex cover problem has a half integer solution.*

Proof. Assume that there is no half integer solution, then pick $\boldsymbol{x} = (x_v)_{v \in V}$ to be an optimal solution to equation 3 with the most half integer x_v (i.e., $\#\big\{x_v : x_v \in \{0, 0.5, 1\}\big\}$

[1] Notice that $1/(1 - \epsilon) = 1 + \mathcal{O}(\epsilon)$ when ϵ small.

is maximized), consider $\boldsymbol{x}^+ = (x_v^+)_{v\in V}, \boldsymbol{x}^- = (x_v^-)_{v\in V}$, where

$$x_v^{\pm} = \begin{cases} x_v, & \text{If } x_v \in \{0, 0.5, 1\} \\ x_v \pm \epsilon, & \text{If } 0 < x_v < 0.5 \\ x_v \mp \epsilon, & \text{If } 0.5 < x_v < 1 \end{cases}$$

It could be easily seen that $\boldsymbol{x}^{\pm}$ are two solutions to equation 3 if ϵ small enough. Notice that $(\boldsymbol{w}^\intercal \boldsymbol{x}^+ + \boldsymbol{w}^\intercal \boldsymbol{x}^-)/2 = \boldsymbol{w}(\boldsymbol{x}^+ + \boldsymbol{x}^-)/2 = \boldsymbol{w}^\intercal \boldsymbol{x}$, so one of $\boldsymbol{x}^{\pm}$ is a solution no worse than $\boldsymbol{x}$. Choose a suitable ϵ then we will obtain an solution with more half integer entries which leads to a contradiction. □

Next we prove a simple lemma:

Lemma 5. *Given a proper k-coloring of the graph, then:*

(1) $V_t = \{v : v \text{ is not colored } t\}$ is a vertex cover.

(2) There is a vertex cover with cost (total weight) no more then $W(k-1)/k$, where $W \triangleq \sum_{v\in V} w_v$ is the sum of the weight of all the vertices.

Proof. If V_t doesn't cover $e = (u, v)$, then we must have both u, v is colored t, which is impossible since it is a proper coloring. This proves (1).

To prove (2), Notice that

$$\sum_{1\le t\le k} \sum_{v\in V_t} w_v = k \sum_{v\in V} w_v - \sum_{v\in V} w_v = (k-1)W$$

Thus the some of cost of these k vertex covers is $(k-1)W$, so one vertex cover must have cost no more then the averge, $W(k-1)/k$. □

Now, let $\boldsymbol{x}$ be an half integer optimal solution of equation 3. Define $V_\alpha \triangleq \{v \in V : x_v = \alpha\}$, then $V = V_0 \cup V_{0.5} \cup V_1$. Let $W_{0.5} = \sum_{v\in V_{0.5}} w_v$, by lemma 5, there is a vertex cover C with cost no more then $W_{0.5}(k-1)/k$. We claim that $C \cup V_1$ is a vertex cover, since if it does not cover an edge $e = (u, v)$, then either u, v are both in V_0, or one of them is in V_0 and the other is in $V_{0.5}$. But both cases are impossible, since then $\boldsymbol{x}$ won't satisfied $x_u + x_v \ge 1$. Now, let S_C be the cost of this vertex cover, S_{OPT} be the cost of the optimal vertex cover, and $S_{\text{LP}} \triangleq \boldsymbol{w}^\intercal \boldsymbol{x}$ be the optimal cost of $\boldsymbol{x}$ in the relaxed LP, then

$$\Big(2 - \frac{2}{k}\Big) S_{\text{OPT}} \overset{(1)}{\ge} 2\frac{k-1}{k} S_{\text{LP}} \overset{(2)}{\ge} 2\frac{k-1}{k} \sum_{v\in V_{0.5}} 0.5 w_v + \sum_{v\in V_1} w_v \ge \frac{k-1}{k} W + \sum_{v\in V_1} w_v \ge S_C$$

Where (1) is because the optimal vertex cover is also a solution to the relaxed LP, and (2) is because $k \ge 2 \implies 2(k-1)/k \ge 1$.

Collaborators: None.

4 Problem 4.

Consider the following LP formulation:

$$
\begin{array}{llrcll}
\text{minimize} & C & & & & \text{(The maximum congestion)} \\
\text{s.t.} & \displaystyle\sum_{i=1}^{k} f_i(g_i(u,v)+g_i(v,u)) & \leq & C, & \forall u,v \in V & \text{(C is the maximum congestion)} \\
\text{and } \forall i, & g_i(u,v) & = & 0, & \forall (u,v) \notin E & \text{(Edge contraints)} \\
& \displaystyle\sum_v g_i(u,v) - g_i(v,u) & = & 0, & \forall u \in V \setminus \{s_i, t_i\} & \text{(Flow conservation)} \\
& \displaystyle\sum_v g_i(s_i,v) - g_i(v,s_i) & = & 1, & & \text{(Flow conservation at s_i)} \\
& \displaystyle\sum_v g_i(t_i,v) - g_i(u,t_i) & = & -1, & & \text{(Flow conservation at t_i)} \\
& f_i(u,v) & \geq & 0, & \forall u,v \in V & \text{(Each flow is non-negative)}
\end{array} \tag{2}
$$

Notice that each g_i would be a valid s_i-t_i flow with flow 1.

We now give the following lemma:

Lemma 6. *If g is an s-t flow with size $|g|$. Let $P = \{p_1, p_2, \cdots, p_n\}$ be all the simple paths from s to t, then g could be decomposed to $g = a_1q_1 + a_2q_2 + \cdots + a_nq_n + \mathcal{C}$, where each q_i is an s-t flow along path p_i with size 1, $\mathcal{C}$ is a circular flow (i.e., feasible flow without s, t) and $\sum a_i = |g|$.*

Proof. Assume that $|g| > 0$. Consider the graph $G' = (V, E')$ such that $(u,v) \in E' \iff g(u,v) > 0$. If there is a simple cycle $\mathcal{C}$ in this graph, then we could subtract q flow from each edge in C, where $q = \min_{e \in C} g(e)$, then the new flow network is still a valid flow with equal size, and the cycle disappeared in C. Since there are only finite simple cycle, we could assume that G' doesn't have loop (The circular flow decomposed in this stage would be collected in $\mathcal{C}$).

Now, we claim that s could reach t in G'. If not, let S be all the vertices that could be reached by s, and $T \triangleq V \setminus S$, then $\sum_{u \in S, v \in T} g(u,v) = 0$. Then by the conservation of flow,

$$0 \geq \sum_{u \in S, v \in T} g(u,v) - g(v,u) = \sum_{u \in S, v \in V} g(u,v) - g(v,u) = \sum_{v \in V} g(s,v) - g(v,s) = |g| > 0$$

which leads to a contradiction. Thus there is a simple path, say p_1, such that each edge on p_1 has positive flow. Let $a_1 \triangleq \min_{e \in p_1} g(e)$, and consider $g' \leftarrow g - aq_1$. Then p_1 disappeared in the corresponding graph G'. Repeat this process, and since there are only finite paths, eventually the procedure ended. And by the fact that G' has no cycle and the conservation of flow, we know that $f(u,v) = 0$ for all u, v and the decomposition is completed. $\square$

By lemma 6 we also know that if we restrict that $g_i(u,v) \in \{0,1\}, \forall i,u,v$ in linear programming (2), then each flow g_i is a single path from s_i to t_i (if there is circular flow, then remove these flow gives a better solution). Thus the problem in the statement correspond to the ILP of (2). And if the solution of the ILP is C_{ILP}, and the solution of the LP is C_{LP}, then $C_{\mathrm{ILP}} \geq C_{\mathrm{LP}}$.

Now we need another lemma:

Lemma 7. *If there are at most r simple path from s to t, then there is a simple path p such that $\min_{e\in p} g(e) \geq |g|/d$, where g is any s-t flow.*

Proof. By lemma 6, we could decompose $g = a_1q_1 + a_2q_2 + \cdots + a_nq_n + \mathcal{C}$, where each q_i is a flow along a simple path p_i. Since $n \leq d$ and $\sum a_i = |g|$, there is an a_i, say a_1 such that $a_1 \geq |g|/d$. Then every edge (u,v) in p_1 will satisfied $g(u,v) \geq a_1 \geq |g|/d$. Thus p_1 is the desired path. □

Notice that this edge could be easily find in polynomial time. Simply remove edges (u,v) with flow $g(u,v)$ less then $|g|/d$ and preform a DFS from s to t.

Finally we round a solution of LP to an approximation solution of the original problem as following: If $\{g_i\}$ is the solution of the linear programming (2), then let p_i be the path such that each edge in it has flow no less then $|g_i|/r = 1/r$. Then set $\hat{g}_i(u,v) = 1$ if $(u,v) \in p_i$, or else $\hat{g}_i(u,v) = 0$. It is easy to check that $\{\hat{g}_i\}$ is indeed a solution to the original problem. Notice that $\hat{g}_i(u,v)/g_i(u,v) \leq 1/r^{-1} \leq r$, so the congestion of an edge $e = (u,v)$ satisfied

$$\hat{C}(e) = \sum \hat{g}_i(u,v) + \hat{g}_i(v,u) \leq r \sum (g_i(u,v) + g_i(v,u)) = rC_{\mathrm{LP}}(e)$$

Thus

$$\hat{C} \leq rC_{\mathrm{LP}}(e) \leq rC_{\mathrm{ILP}}$$

and hence the rounding gives an r-approximation algorithm which has running time polynomial in $k, |V|, |E|$.

Collaborators: None.

5 Problem 5.

Recall that the corresponding relaxed LP of the non-metric uncapacitated facility location problem is:

$$\begin{array}{llcll} \text{minimize} & \sum c_i y_i + \sum d_{i,j} x_{i,j} & & & \\ \text{s.t.} & \sum_i x_{i,j} & \geq & 1, & \forall j \\ & y_i & \geq & x_{i,j}, & \forall i,j \\ & x_{i,j} & \geq & 0, & \forall i,j \end{array} \tag{3}$$

Now, for a solution $(x_{i,j})_{i,j}, (y_i)_i$ for this LP, we preform a randomize rounding procedure as following:

- For each i, set $\hat{y}_i = 1$ with probability $\min(1, \lambda y_i)$, else set $\hat{y}_i = 0$.
- For each j, set $\hat{x}_{i,j} = 1$ if i is the one with $\hat{y}_i = 1$ and smallest $d_{i,j}$. Also define $\hat{i}_j$ to be this i and $\hat{d}_j \triangleq d_{\hat{i}_j,j}$.

We say this rounding procedure "failed" if no facility is opened or there is a j such that $\hat{d}_j \geq 2D_j$, where $D_j \triangleq \sum_i d_{i,j} x_{i,j}$. Notice that

$$\sum_{i:d_{i,j} \leq 2D_j} x_{i,j} \geq 0.5$$

by Markov inequality, so the event $\hat{d}_j \geq 2D_j$ has probability

$$\prod_{i \in I} (1 - \lambda y_i) \leq \prod_{i \in I} (1 - \lambda x_{i,j}) \leq \exp\left(\sum_{i \in I} -\lambda x_{i,j}\right) \leq \mathrm{e}^{-0.5\lambda}$$

Where $I \triangleq \{i : d_{i,j} \leq 2D_j\}$. has probabililty $\prod(1 - \min(1, \lambda y_i))$. Also if $\hat{d}_j \leq 2D_j$ we must have some facility opened. So by uniform bound, if we choose $\lambda = 2\log n + \log 4$, then

$$\mathbb{P}(\text{failed}) \leq \sum_j \mathbb{P}(\hat{d}_j \geq 2D_j) \leq n\mathrm{e}^{-0.5\lambda} \geq \frac{n}{4n} = \frac{1}{4}$$

Hence the algorithm "success" with probability as least $3/4 = \mathcal{O}(1)$. And in this situation, the expect cost $\mathbf{E}[C]$ is

$$\begin{aligned}
\mathbf{E}[C] &= \mathbf{E}\left[\sum_i \mathbb{P}(\hat{y}_i = 1) c_i\right] + \mathbf{E}\left[\sum_j \hat{d}_j\right] \\
&\leq \sum_i c_i \lambda y_i + \sum_j 2D_j \\
&\leq \mathcal{O}(\log n) \sum_i c_i y_i + \sum_{i,j} d_{i,j} x_{i,j} \\
&\leq \mathcal{O}(\log n) C_{\mathrm{LP}}
\end{aligned}$$

Since $C_{\mathrm{OPT}} \geq C_{\mathrm{LP}}$, randomize rounding produce an $\mathcal{O}(\log n)$ approximation algorithm.

Collaborators: None.