Lab04-Matroid

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

∗ If there is any problem, please contact TA Yiming Liu.
∗ Name: ShaoxiongLin Student ID: 517010910028 Email: Johnson-Lin@sjtu.edu.cn

1. Give a directed graph $G = (V, E)$ whose edges have integer weights. Let $w(e)$ be the weight of edge $e \in E$. We are also given a constraint $f(u) \geq 0$ on the out-degree of each node $u \in V$. Our goal is to find a subset of edges with maximal weight, whose out-degree at any node is no greater than the constraint.

   (a) Please define independent sets and prove that they form a matroid.

   (b) Write an optimal greedy algorithm based on Greedy-MAX in the form of *pseudo code*.

   (c) Analyze the time complexity of your algorithm.

Solution.

(a) We consider the pair $(E, \mathcal{I})$, where E is the edges of graph G, and $\mathcal{I}$ is a family of subsets of E. A subset of E is in $\mathcal{I}$ if its out-degree at any node is no greadter than the constraint. We then prove that $(E, \mathcal{I})$ is matroid.

Suppose $I \in \mathcal{I}$ and $I' \subseteq I$, since $I \in \mathcal{I}$, thus its out-degree at any node is no greater than the constraint, and since $I' \subseteq I$, we can get I' by deleting some edges in I, and this operation will not increase the out-degree of the set, thus I' also satisfies the requirement, so we have $I' \in \mathcal{I}$.

For any subset $F \subseteq E$, let $d_F^+(u)$ be the number of out-edges at u which belong F. Then all maximal independent sets in F have the same size, that is $\sum_{u \in V} \min\{f(u), d_F^+(u)\}$. Thus, we have for any any subset $F \subseteq E$, $u(F) = v(F)$, where $v(F)$ is the maximum size of independent subset in F and $u(F)$ is the minimum size of maximal independent subset in F.

Thus, we can say that $(E, \mathcal{I})$ is a matroid.

(b)

Algorithm 1: *greedy*1

Input: directed graph $G = (V, E)$,$w(e)$ for $e \in E$ and $f(u)$ for $u \in V$

Output: a subset of edges with maximal weight, whose out-degree at any node is no greater than the constraint

sort the edges in G into ordering $w(e_1) \geq w(e_2) \geq \cdots \geq w(e_m)$;
$A \leftarrow \emptyset$;
for $i \leftarrow 1$ *to* m **do**
  if $A \cup \{e_i\} \in \mathcal{I}$ **then**
    $A \leftarrow A \cup \{e_i\}$;
  else
    continue;
return A;

(c) The for loop will run m times, and each time the algorithm needs to judge whether A is still independent after adding e_i into it. If we use an array of size n to store the out-degree of every node in A (if a node is not in A, then the corresponding value in the array is 0), the judgement can be done in $O(n)$ time. Thus, the time complexity of the algorithm is $O(mn)$.

$\square$

2. Let X, Y, Z be three sets. We say two triples (x_1, y_1, z_1) and (x_2, y_2, z_2) in $X \times Y \times Z$ are *disjoint* if $x_1 \neq x_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$. Consider the following problem:

Definition 1 (MAX-3DM). *Given three disjoint sets X, Y, Z and a nonnegative weight function $c(\cdot)$ on all triples in $X \times Y \times Z$,* ***Maximum 3-Dimensional Matching*** *(MAX-3DM) is to find a collection $\mathcal{F}$ of disjoint triples with maximum total weight.*

(a) Let $D = X \times Y \times Z$. Define independent sets for MAX-3DM.

(b) Write a greedy algorithm based on Greedy-MAX in the form of *pseudo code*.

(c) Give a counterexample to show that your Greedy-MAX algorithm in Q. 2b is not optimal.

(d) Show that: $\max\limits_{F \subseteq D} \frac{v(F)}{u(F)} \leq 3$. (Hint: you may need Theorem 1 for this subquestion.)

Theorem 1. *Suppose an independent system $(E, \mathcal{I})$ is the intersection of k matroids $(E, \mathcal{I}_i)$, $1 \leq i \leq k$; that is, $\mathcal{I} = \bigcap_{i=1}^{k} \mathcal{I}_i$. Then $\max\limits_{F \subseteq E} \frac{v(F)}{u(F)} \leq k$, where $v(F)$ is the maximum size of independent subset in F and $u(F)$ is the minimum size of maximal independent subset in F.*

Solution.

(a) We consider the pair $(D, \mathcal{I})$, where $\mathcal{I}$ is a collection of subsets of D. A subset of D is in $\mathcal{I}$ if all triples in it are disjoint.We will prove $(D, \mathcal{I})$ is an independent system. Suppose $I \in \mathcal{I}$ and $I' \subseteq I$, since all triples in I are disjoint and we can get I' by deleting some triples in I, thus we can say all triples in I' are also disjoint, that is $I' \in \mathcal{I}$, so $(D, \mathcal{I})$ is an independent system.

(b)

Algorithm 2: *greedy2*

Input: $D = X \times Y \times Z$ and a nonnegative weight function $c(\cdot)$ on all triples in D

Output: a collection $\mathcal{F}$ of disjoint triples with maximum total weight

```
sort all triples in D into ordering (x_1, y_1, z_1) ≥ (x_2, y_2, z_3) ≥ ··· ≥ (x_n, y_n, z_n);
F ← ∅;
for i ← 1 to n do
    if F ∪ {(x_i, y_i, z_i)} ∈ I then
        F ← F ∪ {(x_i, y_i, z_i)};
    else
        continue;
return F;
```

(c) A possible counterexample is that we let $X = \{1, 2\}, Y = \{3, 4\}, Z = \{5, 6\}$ and suppose there exists a nonnegative weight function $c(\cdot)$ that would give the triples in $X \times Y \times Z$ weights like below. The $\mathcal{F}$ given by Alg.2 is $\{\langle 1, 3, 5\rangle, \langle 2, 4, 6\rangle\}$ and the total weight is 12, however, the optimal solution would give $\mathcal{F}$ as $\{\langle 1, 4, 6\rangle, \langle 2, 3, 5\rangle\}$, the total weight is 14.

triple	weight
$\langle 1,3,5\rangle$	10
$\langle 1,3,6\rangle$	9
$\langle 1,4,5\rangle$	8
$\langle 1,4,6\rangle$	7
$\langle 2,3,5\rangle$	7
$\langle 2,3,6\rangle$	5
$\langle 2,4,5\rangle$	4
$\langle 2,4,6\rangle$	2

(d) Let $\mathcal{I}_x$ be a collection of subsets of D, a subset $I \subseteq D$ is in $\mathcal{I}_x$ if for any two different triples in it, say $\langle x_i, y_i, z_i\rangle$ and $\langle x_j, y_j, z_j\rangle$, we have $x_i \neq x_j$. $\mathcal{I}_y$ and $\mathcal{I}_z$ are similar. Then, we can see that $\mathcal{I} = \bigcap_{i\in\{x,y,z\}} \mathcal{I}_i$, where $\mathcal{I}$ is the same as the one defined in the answer to Q. 2a.

We want to prove that $(D, \mathcal{I}_i)$ $(i \in \{x, y, z\})$ is a matroid. Suppose $I \in \mathcal{I}_x, I' \subseteq I$, it is easy to see $I' \in \mathcal{I}_x$. Suppose there are two different subsets of D, say I_1 and I_2, such that $I_1 \in \mathcal{I}_x, I_2 \in \mathcal{I}_x$ and $|I_1| < |I_2|$. We say that there must exists a triple $\langle x_{i2}, y_{i2}, z_{i2}\rangle$ in I_2, such that x_{i2} is different from the first element of every triple in I_1. Suppose it is not that case, then we have for any triples in I_2, there exists a triple in I_1, who has the same first element with it. Then according to the definition of $\mathcal{I}_x$, we have that $|I_1| \geq |I_2|$, which contradicts that $|I_1| < |I_2|$. Thus, there exist a triple $\langle x_{i2}, y_{i2}, z_{i2}\rangle$, whose first element is different from the first element of every triple in I_1, then $I_1 \cup \{\langle x_{i2}, y_{i2}, z_{i2}\rangle\}$ is in $\mathcal{I}_x$. Thus, $(D, \mathcal{I}_x)$ is a matroid, and the prove for $(D, \mathcal{I}_y)$ and $(D, \mathcal{I}_z)$ are similar.

Therefore, according to **theorem**1, we have $\max\limits_{F\subseteq D} \frac{v(F)}{u(F)} \leq 3$.

□

3. **Crowdsourcing** is the process of obtaining needed services, ideas, or content by soliciting contributions from a large group of people, especially an online community. Suppose you want to form a team to complete a crowdsourcing task, and there are n individuals to choose from. Each person p_i can contribute v_i $(v_i > 0)$ to the team, but he/she can only work with up to c_i other people. Now it is up to you to choose a certain group of people and maximize their total contributions $(\sum_i v_i)$.

   (a) Given $\mathrm{OPT}(i, b, c)$ = maximum contributions when choosing from $\{p_1, p_2, \cdots, p_i\}$ with b persons from $\{p_{i+1}, p_{i+2}, \cdots, p_n\}$ already on board and at most c seats left before any of the existing team members gets uncomfortable. Describe the optimal substructure as we did in class and write a recurrence for $\mathrm{OPT}(i, b, c)$.

   (b) Design an algorithm to form your team using dynamic programming, in the form of *pseudo code.*

   (c) Analyze the time and space complexities of your design.

Solution.

(a) We consider whether we can shoose p_i. If we can choose p_i, then

$$\mathrm{OPT}(i, b, c) = \max\{\mathrm{OPT}(i-1, b, c), v_i + \mathrm{OPT}(i-1, b+1, \min\{c-1, c_i\})\}$$

If we can not choose p_i, then

$$\mathrm{OPT}(i, b, c) = \mathrm{OPT}(i-1, b, c)$$

Thus, the recurrence for $\mathrm{OPT}(i,b,c)$ is

$$\mathrm{OPT}(i,b,c)=\begin{cases}\max\{\mathrm{OPT}(i-1,b,c), v_i+\mathrm{OPT}(i-1,b+1,\min\{c-1,c_i\})\} & c_i\geq b, c>0, \\ & i\geq 0 \\ \mathrm{OPT}(i-1,b,c) & c_i<b, c>0, \\ & i\geq 0 \\ 0 & \text{otherwise}\end{cases}$$

(b) We first creat an 3 dimensional array M and initialize it with -1 and two global arrays, which are initialized by c_i and v_i ($i \in \{i, 2, \ldots, n\}$).

Algorithm 3: Compute-M(i,b,c)

Input: i, b and c
Output: the value of $M[i,b,c]$

if $M[i,b,c]$ *has been computed* **then**
  return $M[i,b,c]$;
else if $c_i \geq b$ *and* $c > 0$*and* $i \geq 0$ **then**
  $M[i,b,c] \leftarrow$
   $\max\{\text{Compute-M}(i-1,b,c), v_i + \text{Compute-M}(i-1,b+1,\min\{c-1,c_i\})\}$
else if $c_i < b$ *and* $c > 0$ *and* $i \geq 0$ **then**
  $M[i,b,c] \leftarrow \text{Compute-M}(i-1,b,c)$
else
  $M[i,b,c] \leftarrow 0$
return $M[i,b,c]$;

Suppose the array M has been computed, then we call Find-Solution$(n,0,n)$ to get the people we choose.

Algorithm 4: Find-Solution(i,b,c)

---

if $i = 0$ **then**
  return $\emptyset$;
else if $v_i + M[i-1,b+1,\min\{c-1,c_i\}] > M[i-1,b,c]$ **then**
  return $\{i\} \cup$ *Find-Solution*$(i-1,b+1,\min\{c-1,c_i\})$;
else
  return *Find-Solution*$(i-1,b,c)$;

(c) Since we need an 3 dimensional array M and two global arrays, the space complexities is $O(n^3)$. To choose the people we need, we just need to figure out the aarray M, time spent on this is no more than the number of items in M, after that we need to call Find-Solution$(n,0,n)$, this takes $O(n)$. Thus, time complexities is bounded by $O(n^3)$.

□

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.