Algorithm Design and Analysis

Assignment 5

Deadline: Dec 18, 2022

1. (25 points) **[Fair Division Revisited]** Suppose we want to allocate m indivisible items to n agents. A $n \times m$ matrix $V = [v_{ij}]_{i=1,\ldots,n;j=1,\ldots,m}$ describes agents' preferences over the items, where v_{ij} is the value of item j according to agent i's preference. We assume agents' valuations are binary, i.e., $v_{ij} \in \{0, 1\}$ for any i and j. Given a subset S of items, agent i's value on S is naturally defined by $v_i(S) = \sum_{j \in S} v_{ij}$.

   An allocation $(A_1, \ldots, A_n)$ is a partition of $[m] := \{1, \ldots, m\}$, where $A_i \subseteq [m]$ is the set of the items allocated to agent i. An allocation is *proportional* if $v_i(A_i) \geq \lfloor \frac{1}{n} v_i([m]) \rfloor$ holds for any agent i. That is, in a proportional allocation, each agent receives his/her average integer value, rounded down, of the entire item set. For example, if $v_{ij} = 1$ for all i and j, the average value for each agent is m/n, and each agent needs to get at least $\lfloor \frac{m}{n} \rfloor$ items to guarantee proportionality.

   In this question, we are going to prove the existence of proportional allocations and design an algorithm to find one.

   (a) (5 points) Let us first assume, for this moment, that items are *divisible*. That is, we are allow to fractionally allocate a single item such that an agent gets some portion of it. Show that there exists a *fractional* allocation such that each agent i receives a value of at least $\frac{1}{n} v_i([m])$.

   (b) (5 points) Consider the flow network with $m+n+2$ vertices $s, u_1, \ldots, u_n, w_1, \ldots, w_m, t$ described as follows. There is a directed edge from s to each u_i with capacity $\frac{1}{n} v_i([m])$. There is a directed edge from u_i to w_j with capacity ∞ if $v_{ij} = 1$. There is a directed edge from each w_j to t with capacity 1. According to Part (a), what is the maximum flow in this network?

   (c) (5 points) Now, we modify the network by reducing the capacity of each edge (s, u_i) from $\frac{1}{n} v_i([m])$ to $\lfloor \frac{1}{n} v_i([m]) \rfloor$. What is the maximum flow in this network?

   (d) (5 points) Show that the maximum flow in Part (c) can be made integral. Use this result to show that the proportional allocation always exists even when items are *indivisible*.

   (e) (5 points) Design an algorithm to compute a proportional allocation.

2. (25 points) **[Hall Marriage Theorem]** Given a bipartite graph $G = (A, B, E)$ with $|A| = |B| = n$, *Hall Marriage Theorem* states that G contains a perfect matching (i.e., a matching with size n) if and only if $|N(S)| \geq |S|$ for any $S \subseteq A$, where $N(S) = \{b \in B \mid \exists a \in S \subseteq A : (a, b) \in E\}$ is the set of all the neighbors of the vertices in S. In this question, we are going to prove Hall Marriage Theorem.

   (a) (5 points) Prove the necessity part: if G contains a perfect matching, then $|N(S)| \geq |S|$ for any $S \subseteq A$.

   (b) (10 points) Construct a directed weighted graph with vertex set $V = A \cup B \cup \{s\} \cup \{t\}$. The edge set is defined as follows. For each $a \in A$, construct a directed edge (s, a) with weight 1; for each $a \in A$ and $b \in B$, construct a directed edge (a, b) with weight ∞ if (a, b) is an edge in the original graph G; for each $b \in B$, construct a directed edge (b, t) with weight 1. Prove that, if $|N(S)| \geq |S|$, the minimum s-t cut has value n.

   (c) (10 points) Prove the sufficiency part: if $|N(S)| \geq |S|$ holds for all $S \subseteq A$, then G contains a perfect matching.

3. (25 points) [**LP-Duality and Total Unimodularity**] In this question, we will prove König-Egerváry Theorem, which states that, in any bipartite graph, the size of the maximum matching equals to the size of the minimum vertex cover. Let $G = (V, E)$ be a bipartite graph.

   (a) (4 points) Explain that the following linear program describes *the fractional version of* the maximum matching problem (i.e., replacing $x_e \geq 0$ to $x_e \in \{0, 1\}$ gives you the exact description of the maximum matching problem).

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{e : e=(u,v)} x_e \leq 1 && (\forall v \in V) \\ & x_e \geq 0 && (\forall e \in E) \end{aligned}$$

   (b) (4 points) Write down the dual of the above linear program, and justify that the dual program describes the fractional version of the minimum vertex cover problem.

   (c) (7 points) Show by induction that the *incident matrix* of a bipartite graph is totally unimodular. (Given an undirected graph $G = (V, E)$, the incident matrix A is a $|V| \times |E|$ zero-one matrix where $a_{ij} = 1$ if and only if the i-th vertex and the j-th edge are incident.)

   (d) (6 points) Use results in (a), (b) and (c) to prove König-Egerváry Theorem.

   (e) (4 points) Give a counterexample to show that the claim in König-Egerváry Theorem fails if the graph is not bipartite.

4. (25 points) **[LP and Vertex Cover]** In this question, we are going to design a 2-approximation algorithm for the minimum vertex cover problem that is different from the one learned in the class. Let $G = (V, E)$ be an undirected graph for which we want to find a minimum vertex cover.

   (a) (5 points) Explain that the following linear program describes *the fractional version* of the vertex cover problem (i.e., replacing $x_u \geq 0$ to $x_u \in \{0, 1\}$ gives you the exact description of the minimum vertex cover problem).

$$\begin{aligned} \text{minimize} \quad & \sum_{u \in V} x_u & \\ \text{subject to} \quad & x_u + x_v \geq 1 & (\forall (u, v) \in E) \\ & x_u \geq 0 & (\forall u \in V) \end{aligned}$$

   (b) (5 points) Let $\{x_u^*\}_{u \in V}$ be an optimal solution to the linear program in Part (a). Prove that $\sum_{u \in V} x_u^*$ is a lower bound to the size of any vertex cover.

   (c) (10 points) Consider the following algorithm.

   - Solve the linear program in Part (a) and obtain an optimal solution $\{x_u^*\}_{u \in V}$;
   - Return $S = \{u \in V \mid x_u^* \geq 0.5\}$.

   Prove that the algorithm returns a valid vertex cover, and it is a 2-approximation algorithm.

   (d) (5 points) Show that there exists an integral optimal solution to the linear program in Part (a) if G is a bipartite graph. Notice that an integral optimal solution to this linear program directly gives you a minimum vertex cover. (Hint: This almost straightforwardly follows from one of the previous questions you have solved. Can you see it?)

5. (Bonus 60 points) **[Dinic's Algorithm]** In this question, we are going to work out *Dinic's algorithm* for computing a maximum flow. Similar to Edmonds-Karp algorithm, in each iteration of Dinic's algorithm, we update the flow f by increasing its value and then update the residual network G^f. However, in Dinic's algorithm, we push flow along *multiple* s-t paths in the residual network instead of a *single* s-t path as it is in Edmonds-Karp algorithm.

   In each iteration of the algorithm, we find the *level graph* of the residual network G^f. Given a graph G with a source vertex s, its level graph $\overline{G}$ is defined by removing edges from G such that only edges pointing from level i to level $i+1$ are kept, where vertices at level i are those vertices at distance i from the source s. An example of level graph is shown in Fig. 1.

   Next, the algorithm finds a *blocking flow* on the level graph $\overline{G}^f$ of the residual network G^f. A blocking flow f in G is a flow such that each s-t path contains at least one *critical edge*. Recall that an edge e is *critical* if the amount of flow on it reaches its capacity: $f(e) = c(e)$. Fig. 2 gives examples for blocking flow.

   Dinic's algorithm is then described as follows.

   1. Initialize f to be the empty flow, and $G^f = G$.
   2. Do the following until there is no s-t path in G^f:
      - construct the level graph $\overline{G}^f$ of G^f.
      - find a blocking flow on $\overline{G}^f$.
      - Update f by adding the blocking flow to it, and update G^f.

   Complete the analysis of Dinic's algorithm by solving the following questions.

   (a) (15 points) Prove that, after each iteration of Dinic's algorithm, the distance from s to t in G^f is increased by at least 1.

   (b) (15 points) Design an $O(|V| \cdot |E|)$ time algorithm to compute a blocking flow on a level graph.

   (c) (10 points) Show that the overall time complexity for Dinic's algorithm is $O(|V|^2 \cdot |E|)$.

   (d) (20 points) **(challenging)** We have seen in the class that the problem of finding a maximum matching on a bipartite graph can be converted to the maximum flow problem. Show that Dinic's algorithm applied to finding a maximum matching on a bipartite graph only requires time complexity $O(|E| \cdot \sqrt{|V|})$.

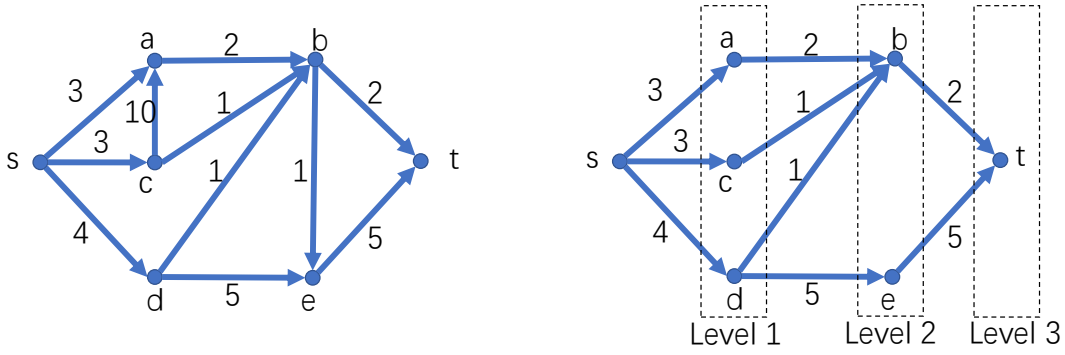

Figure 1: The graph shown on the right-hand side is the level graph of the graph on the left-hand side. Only edges pointing to the next levels are kept. For example, the edges (c, a) and (b, e) are removed, as they point at vertices at the same level. If there were edges pointing at previous levels, they should also be removed.

6. How long does it take you to finish the assignment (including thinking and discussing)? Give a score (1,2,3,4,5) to the difficulty. Do you have any collaborators? Write down their names here.

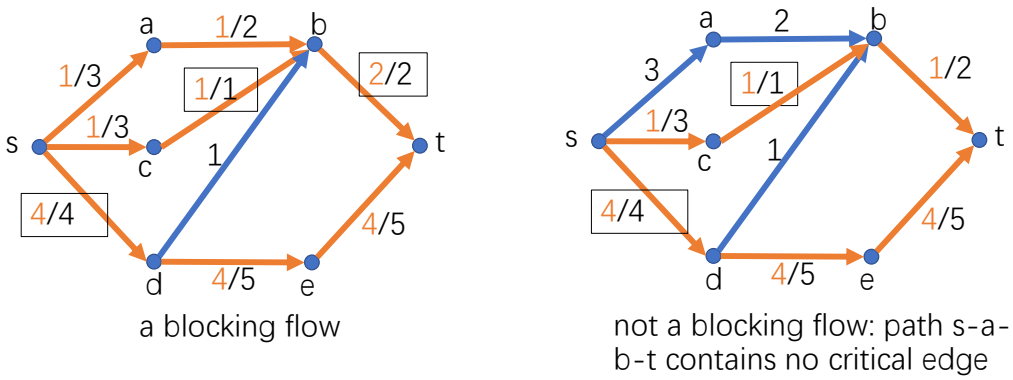

Figure 2: Blocking flow examples