Bin Packing

Suppose that we are given a set of n objects, where the size s_i of the i-th object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1. Note that the problem of determining the minimum number of bins required is NP-hard.

The first-fit heuristic takes each object in turn and places it into the first bin that can accommodate it. Let $S = \sum_{i=1}^{n} s_i$.

1. Argue that the optimal number of bins required is at least $\lceil S \rceil$ (6 points).
2. Argue that the first-fit heuristic leaves at most one bin less than half full (6 points).
3. Prove that the number of bins used by the first-fit heuristic is never more than $\lceil 2S \rceil$ (6 points).
4. Prove an approximation ratio of 2 for the first-fit heuristic (6 points).

Multiagent Weighted Vertex Cover

The input consists of an undirected graph $G := (V, E)$ and an agent set $A := \{ 1, \ldots, k \}$. An agent i has a cost function $c_i : V \to \mathbb{R}_{\geq 0}$, which indicates agents i's preference for each vertex. A feasible solution $S \subseteq V$ is a subset of vertices such that for each edge $(u, v) \in E$, either $u \in S$ or $v \in S$. The goal is to choose a feasible solution S such that $\max_{i \in A} \sum_{v \in S} c_i(v)$ is minimized. Note that the problem reduces to the classical weighted vertex cover problem when there is only one agent, i.e., $|A| = 1$. Recall that the classical weighted vertex cover problem admits a 2-approximate algorithm.

Algorithm. Consider the following simple algorithm: define a merged cost function $\hat{c} := \sum_{i=1}^{k} c_i$, i.e., for each $v \in V$, $\hat{c}(v) := \sum_{i=1}^{k} c_i(v)$. Now, we call the 2-approximate algorithm for classical weighted vertex cover based on the cost function $\hat{c}$; the algorithm shall output a feasible solution S. We just return S as the solution to the multiagent weighted vertex cover problem.

(a) (20 marks) Prove that the above algorithm is a $2k$-approximate algorithm.

Scheduling with Testing

Consider the following scheduling problem on a single machine. There is a job set $J := \{ 1, \ldots, n \}$ consisting of n jobs. Each job $j \in J$ has two possible processing times: upper processing time $\hat{p}_j \in \mathbb{R}_{\geq 0}$ and lower processing time $\check{p}_j \in \mathbb{R}_{\geq 0}$. Note that $0 \leq \check{p}_j \leq \hat{p}_j$ for all $j \in J$. If a job j is tested, it will take $\check{p}_j$ to execute; otherwise, its processing time is $\hat{p}_j$. We can only test $k \leq n$ jobs (k is an integer). The goal is to choose k jobs to test such that the makespan is minimized, where makespan is defined as the maximum completion time among all jobs.

Example

Suppose that we have 3 jobs $J := \{ 1, 2, 3 \}$. For job 1, we have $\check{p}_1 = 2$ and $\hat{p}_1 = 3$. For job 2, we have $\check{p}_2 = 2.5$ and $\hat{p}_2 = 5$. For job 3, we have $\check{p}_3 = 0$ and $\hat{p}_3 = 10$. We can only test one job. If we test job j_3, the completion time of job 3 will be 0; the makespan 1 and 2 is 8. If we test job j_1, the makespan is 17.

Q1. (5 points) Suppose that $\check{p}_j = 0$ for all $j \in J$. Which k jobs should be tested such that the makespan is minimized?

Q2. (5 points) Now, we remove the assumption of **Q1**, i.e., $\check{p}_j, \hat{p}_j \in \mathbb{R}_{\geq 0}$ and $\check{p}_j \leq \hat{p}_j$ for all $j \in J$. Which k jobs should be tested such that the makespan is minimized?

Q3. (10 points) Consider the following setting in which the algorithm does not know the value of $\check{p}_j$ for each job j. All other pieces of information are known by the algorithm. Design an algorithm such that the algorithm is 2-approximation and prove the correctness.