CSC2420 - Algorithm Design, Analysis and Theory **Fall 2022**

Lecture 2: Approximation Algorithms Review

Lecturer: Allan Borodin *Scribe: Kevin Gao*

2.1 Online Bipartite Matching

In a graph $G = (V, E)$, a matching is $M = (V, E')$ where $E' \subseteq E$ such that for every vertex v, the degree of v, $\deg_M(v)$ is at most 1. In an unweighted graph, the goal is to maximize the number of edges in the match. In an edge-weighted graph, the objective is to maximize the sum of weights of edges in the match. In the offline setting, the currently best result is Jack Edmonds's polynomial algorithm.

In a bipartite graph $G = (U \cup V, E)$, the vertices are partitioned into two sets U, V with $U \cap V = \emptyset$ and $E \subseteq U \times V$.

In terms of online algorithms, the most common online model is one where one set of the vertices V are known in advance, and the other isde of hte vertices U, arrive **online along with their adjacent edges**. The natural greedy approach would be to admit an edge whenever possible. This gives an 1/2-approximation algorithm. This is the best deterministic algorithm.

There exists a randomized algorithm that achieves the competitive ratio $1 - 1/e$. The algorithm, known as the **Ranking Algorithm**, works by randomly assigning priorities to the offline vertices. Then when the online vertices arrive, we match them to the available offline vertex with the highest priority.

Almost all the optimization problems we study can be formulated as **IPs**. These IPs can be relaxed to LPs. For example, the bipartite matching problem can be formulated as

$$\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{(u_i,v_j)\in E} x_{ij} & \\
\text{subject to} & \displaystyle\sum_{j:(u_i,v_j)\in E} x_{i,j} \le 1 & u_i \in V \\
& \displaystyle\sum_{i:(v_i,v_j)\in E} x_{i,j} \le 1 & v_j \in V \\
& x_{i,j} \ge 0 & (u_i, v_j) \in E
\end{array}$$

The Ranking Algorithm can also be viewed as a deterministic algorithm in the ROM (random order model) model. However, it is not known if the $1 - 1/e$ competitive ratio is optimal under the ROM model.

2.1.1 AdWords

In AdWords, edge weights represent bids b_{ij} by an offline advertiser v_j for a query impression u_i. Each offline advertiser v_j has a budget B_j. The objective is to find a matching M so as to maximize the following objective:

$$\sum_j \min\left\{B_j, \sum_{i:e_{ij}\in M} b_{ij}\right\}$$

i.e. Each advertiser does not generate mmore revenue than its budget.

2.2 Makespan

2.2.1 Partial Enumeration Greedy

Recall the makespan problem and the **LPT (longest processing time)** approximation algorithm. We can improve the $\frac{4}{3} - \frac{1}{3m}$ competitive ratio by solving a portion of the problem using brute-force search.

Definition 2.1. *Optimally schedule the largest k jobs (for $0 \leq k \leq n$) and then greedily schedule the remaining jobs in any order.*

The algorithm has approximation ratio is no worse than $(1 + \frac{1-1/m}{1+\lfloor k/m \rfloor})$. The running time is $O(m^k + n \log n)$. Setting $k = (1-\epsilon)/\epsilon$ gives a ratio of at most $(1+\epsilon)$. This is a PTAS with time $O(m^{m/\epsilon} + n \log n)$.

Online is not necessarily greedy. The best known approximation ratio is 1.9201 (Fleischer and Wahl) and there is 1.88 inapproximation bound (Rudin). Basic idea of some of the better algorithm is to leave some room for a possible large job; this forces the online algorithm to be **non-greedy** in some sense but still within the online model.

2.2.2 Restricted Machine Model

In the **restricted machine model**, every job J_j is described by a pair (p_j, S_j) where $S_j \subseteq \{1, \ldots, m\}$ is the set of machines on which J_j can be scheduled. If each job has two allowable machine, this is equivalent to the graph orientation problem.

Azar et al. showed that $\log_2(m)$ is the best competitive ratio for deterministic online algorithms with the upper bounds obtained by the "natural greedy algorithm". For randomized online, the ratio is $\ln(m)$.

2.2.3 Unrelated Machine Model

This is the most general case of the makespan machine models. Under this model, a job J_i is represented by a vector $(p_{j,1}, \ldots, p_{j,m})$ where $p_{j,i}$ is the time to process job J_j on machine i. There is an online algorithm with approximation $O(\log m)$. This is currently the best-known approximation for greedy-like (priority-based) algorithms even for the restricted machines model.

2.2.4 Precedence Constraints

Graham also considered the makespan problem on identical machines for jobs satisfying a precedence constraint. Suppose that $\prec$ is a partial ordeer on jobs. If $J_i \prec J_k$, then J_i must complete before J_k can be started. Assuming jobs are ordered to respect the partial order, Graham showed that the ratio $2 - 1/m$ is achieved by the "natural greedy algorithm", call it $\mathcal{G}_{\prec}$.

Counterintuitively, many seemingly beneficial relaxations could actually lead to increase in the objective function that we want to minimize in makespan:

- relaxing the precedence $\prec$

- decreasing the processing time of certain jobs
- adding more machines

2.3 Knapsack

Definition 2.2 (Knapsack problem)**.**
***Input**: Knapsack size capacity C and n items $\mathcal{I} = \{I_1, \ldots, I_n\}$ where $I_j = (v_j, s_j)$. v is the profit and s is the size of the item.*
***Output**: A feasible subset $S \subseteq \{1, \ldots, n\}$ satisfying $\sum_{j \in S} s_j \leq C$ that maximizes $V(S) = \sum_{j \in S} v_j$.*

For some problems, approximation ratio is denoted with a value ≥ 1. Many people (and literatures) use approximation ratios $\rho \leq 1$ for maximization problems. In this case, ALG $\geq \rho$OPT.

For the purpose of the knapsack problem, we generally restrict ourselves to the offline setting. The algorithm could still be "myopic". There have been studies done on the online version of the problem.

The partial enumeration greedy algorithm:

```
PARTIAL-ENUM-KNAPSACK($\mathcal{I} = \{(v_1, s_1), \ldots, (v_n, s_n)\}$)
1  sort $\mathcal{I}$ in non-increasing order by $v_i/s_i$
2  for every feasible subset $H \subseteq I$ with $|H| \leq k$
3      $R = \mathcal{I} - H$
4      $\text{OPT}_H$ = optimal solution for $H$
5      consider items in $R$ in order of $v_i/s_i$
6      greedily add items to $\text{OPT}_H$ not exceeding knapsack capacity $C$
7  return $\text{OPT}_H$ with max profit
```

Theorem 2.3 (Sahni 1975)**.** $V(\text{OPT}) \leq (1 + 1/k)V(\text{PARTIAL-ENUM-KNAPSACK})$. *The algorithms runs in kn^k time and setting $k = 1/\epsilon$ gives an $(1 + \epsilon)$ approximation algorithm.*

2.4 Priority Algorithm Model

What is the intuitive nature of a greedy algorithm? With the exception of Huffman coding, all these algorithms consider one input item in each iteration and make an **irrevocable greedy decision** about that item.

```
1   $\mathcal{J}$ = set of all possible inputs
2   $\preceq$ = a total ordering on $\mathcal{J}$ (typically induced by a function $f$)
3   $\mathcal{I} \subset \mathcal{J}$ = actual input to the algorithm
4   $S = \emptyset$                              // items already examined by the algorithm
5   $i = 0$
6   while $\mathcal{I} - S \neq \emptyset$
7       $i = i + 1$
8       $\mathcal{I} = \mathcal{I} - S$
9       $I_i = \min_{\preceq}\{I \in \mathcal{I}\}$        // select min element based on the ordering $\preceq$
10      make an irrevocable decision $D_i$ concerning $I_i$
11      $S = S \cup \{I_i\}$
```

Here, we make no assumption on the complexity or even the computability of the ordering $\preceq$ or function f.

This template can be modified to allow the algorithm to determine the total ordering dynamically based on the elements it has observed so far.

```
$\mathcal{J}$ = set of all possible inputs
$\mathcal{I} \subset \mathcal{J}$ = actual input to the algorithm
$S = \emptyset$                       // items already examined by the algorithm
$i = 0$
while $\mathcal{I} - S \neq \emptyset$
    $i = i + 1$
    $\preceq_i$ = a total ordering on $\mathcal{J}$ (typically induced by a function $f_i$)
    $\mathcal{I} = \mathcal{I} - S$
    $I_i = \min_{\preceq_i}\{I \in \mathcal{I}\}$      // select min element based on the ordering $\preceq_i$
    make an irrevocable decision $D_i$ concerning $I_i$
    $S = S \cup \{I_i\}$
    $\mathcal{J} = \mathcal{J} - \{I \in \mathcal{I} \mid I \preceq_i I_i\}$
```

2.4.1 Inapproximation w.r.t. Priority Model