Lecture 6: Greedy II

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

September 26, 2023

Overview

- Knapsack Problems
- Scheduling to minimize lateness
- Acknowledgements

0-1 Knapsack problem

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a subset of the items $S \subseteq [n]$ such that:
  1. $\sum_{k \in S} w_i \leq L$ (respect max load)
  2. $\sum_{k \in S} v_i \geq \sum_{i \in T} v_i$ for any other set T that respects max load

0-1 Knapsack problem

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a subset of the items $S \subseteq [n]$ such that:
  1. $\sum_{k \in S} w_i \leq L$ (respect max load)
  2. $\sum_{k \in S} v_i \geq \sum_{i \in T} v_i$ for any other set T that respects max load
- **Model:** Word RAM
- **Situation:** Thief is robbing a store with n items and a bag with load L. The i^{th} item worth v_i moneyz and weighs w_i kgs. Thief wants to take most value possible with these constraints.

0-1 Knapsack problem

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a subset of the items $S \subseteq [n]$ such that:
  1. $\sum_{k \in S} w_i \leq L$ (respect max load)
  2. $\sum_{k \in S} v_i \geq \sum_{i \in T} v_i$ for any other set T that respects max load
- This problem has *optimal substructure property*: if remove an item from optimal solution, say item (v_i, w_i), then remaining load must be optimal for the problem with load $L - w_i$

0-1 Knapsack problem

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a subset of the items $S \subseteq [n]$ such that:
  1. $\sum_{k \in S} w_i \leq L$ (respect max load)
  2. $\sum_{k \in S} v_i \geq \sum_{i \in T} v_i$ for any other set T that respects max load
- This problem has *optimal substructure property*: if remove an item from optimal solution, say item (v_i, w_i), then remaining load must be optimal for the problem with load $L - w_i$
- Can greedy work here?

0-1 Knapsack problem

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a subset of the items $S \subseteq [n]$ such that:
  1. $\sum_{k \in S} w_i \leq L$ (respect max load)
  2. $\sum_{k \in S} v_i \geq \sum_{i \in T} v_i$ for any other set T that respects max load
- This problem has *optimal substructure property*: if remove an item from optimal solution, say item (v_i, w_i), then remaining load must be optimal for the problem with load $L - w_i$
- Can greedy work here?
- Unfortunately doesn't seem to be the case (*NP-hard*) (we will see this problem again and again later in the course...)

Fractional Knapsack

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a list of fractions $(x_1, \ldots, x_n) \in [0,1]^n$ such that:
  1. $\sum_{k \in [n]} x_i w_i \leq L$ (respect max load)
  2. $\sum_{k \in [n]} x_i v_i \geq \sum_{k \in [n]} y_i v_i$ for any list $(y_1, \ldots, y_n)$ respecting max load
- **Model:** Word RAM
- **Situation:** now thief can take fractions of each item.

Fractional Knapsack

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \dots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a list of fractions $(x_1, \dots, x_n) \in [0,1]^n$ such that:
  1. $\sum_{k \in [n]} x_i w_i \leq L$ (respect max load)
  2. $\sum_{k \in [n]} x_i v_i \geq \sum_{k \in [n]} y_i v_i$ for any list $(y_1, \dots, y_n)$ respecting max load
- Problem also has *optimal substructure property*

Fractional Knapsack

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a list of fractions $(x_1, \ldots, x_n) \in [0,1]^n$ such that:
  1. $\sum_{k \in [n]} x_i w_i \leq L$ (respect max load)
  2. $\sum_{k \in [n]} x_i v_i \geq \sum_{k \in [n]} y_i v_i$ for any list $(y_1, \ldots, y_n)$ respecting max load
- Problem also has *optimal substructure property*
- Possible greedy strategy:

Maximize *value per weight*: v_i / w_i

Fractional Knapsack

- **Input:** n items, each with a prescribed value and weight, given by $(v_1, w_1), \ldots, (v_n, w_n)$, as well as a maximum load L
- **Output:** a list of fractions $(x_1, \ldots, x_n) \in [0,1]^n$ such that:
  1. $\sum_{k\in[n]} x_i w_i \leq L$ (respect max load)
  2. $\sum_{k\in[n]} x_i v_i \geq \sum_{k\in[n]} y_i v_i$ for any list $(y_1, \ldots, y_n)$ respecting max load
- Problem also has *optimal substructure property*
- Possible greedy strategy:

  Maximize *value per weight*: v_i/w_i
- Algorithm
  1. Sort items by decreasing order of value per weight
  2. Take as much as possible of item with highest value per weight
  3. Recurse until load is full or no more items

Proof of correctness (fractional case)

- Assuming items are ordered by v_i/w_i in decreasing order
- In fractional case, can assume $\sum_{i\in[n]} x_i w_i = L$

  If $\sum_i w_i \leq L$ then problem is trivial.

Proof of correctness (fractional case)

- Assuming items are ordered by v_i/w_i in decreasing order
- In fractional case, can assume $\sum_{i\in[n]} x_i w_i = L$
- Thus, if $(x_1, \ldots, x_n) \succeq (y_1, \ldots, y_n)$, we have

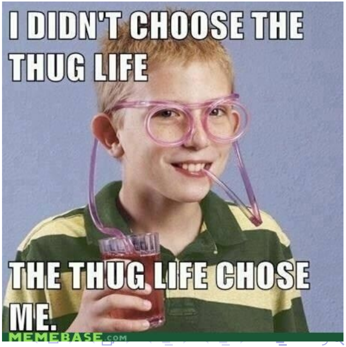
$$\sum_{i\in[n]} (x_i - y_i) v_i = \sum_{k\in[n]} v_i \cdot \left(\sum_{i\le k} (x_i - y_i) \right) \ge 0$$

Proof of correctness (fractional case)

- Assuming items are ordered by v_i/w_i in decreasing order
- In fractional case, can assume $\sum_{i\in[n]} x_i w_i = L$
- Thus, if $(x_1, \ldots, x_n) \succeq (y_1, \ldots, y_n)$, we have

$$\sum_{i\in[n]}(x_i - y_i)v_i = \sum_{k\in[n]} v_i \cdot \left(\sum_{i\leq k}(x_i - y_i)\right) \geq 0$$

- Thug life works!



Why doesn't it work for 0-1 Knapsack?

- Forced to pick entire item, which may prevent you from picking other items
- Counterexample: items $(60, 10), (20, 100), (30, 120)$ and load 50

- Knapsack Problems
- Scheduling to minimize lateness
- Acknowledgements

Scheduling problem strikes back

- **Input:** n tasks with deadlines $(s_1, t_1, d_1), \ldots, (s_n, t_n, d_n)$

  i^{th} task has to be scheduled *on or after* starting time s_i, takes t_i time to complete.

  If i^{th} task scheduled at time T, then *lateness* of i^{th} task defined as $\ell_i = \max\{0, T + t_i - d_i\}$

- **Output:** assignment S of all tasks that minimizes maximum lateness

  $$L(S) := \max_{i \in [n]} \ell_i$$

  so that

  $$L(S) \leq L(S')$$

  for any $S' \neq S$

- **Model:** Word RAM

Scheduling problem strikes back

- **Input:** n tasks with deadlines $(s_1, t_1, d_1), \ldots, (s_n, t_n, d_n)$

  i^{th} task has to be scheduled *on or after* starting time s_i, takes t_i time to complete.

  If i^{th} task scheduled at time T, then *lateness* of i^{th} task defined as $\ell_i = \max\{0, T + t_i - d_i\}$

- **Output:** assignment S of all tasks that minimizes maximum lateness

$$L(S) := \max_{i \in [n]} \ell_i$$

so that

$$L(S) \leq L(S')$$

for any $S' \neq S$

- Without assumptions on starting times, then problem is *NP-hard*...

Scheduling problem strikes back

- **Input:** n tasks with deadlines $(s_1, t_1, d_1), \ldots, (s_n, t_n, d_n)$

  i^{th} task has to be scheduled *on or after* starting time s_i, takes t_i time to complete.

  If i^{th} task scheduled at time T, then *lateness* of i^{th} task defined as
  $$\ell_i = \max\{0, T + t_i - d_i\}$$
- **Output:** assignment S of all tasks that minimizes maximum lateness
  $$L(S) := \max_{i \in [n]} \ell_i$$
  so that
  $$L(S) \leq L(S')$$
  for any $S' \neq S$
- what if we assumed all starting times are the same (say $s_i = 0$)?

Greedy approaches (same starting time)

1. Schedule tasks in order of increasing length

Greedy approaches (same starting time)

1. Schedule tasks in order of increasing length

Ignoring deadlines.

Counterexample: $(1, 100), (10, 10)$

Greedy approaches (same starting time)

2. Schedule tasks in order of increasing *slack time*, i.e. $d_i - t_i$

Greedy approaches (same starting time)

2. Schedule tasks in order of increasing *slack time*, i.e. $d_i - t_i$

Can delay too much easy tasks.
Counterexample: $(1, 2), (10, 10)$

Greedy approaches (same starting time)

3. Sort tasks by increasing order of deadlines, so we can assume $d_1 \leq d_2 \leq \cdots \leq d_n$ and schedule tasks accordingly (i.e. $[n]$). Break ties by scheduling easier task first.

Seems like we are ignoring the times of the tasks...
Should this work?

Earliest Deadline First analysis

- We are assuming that $d_1 \leq d_2 \leq \cdots \leq d_n$
- let $f_0 := 0$ and $f_i := f_{i-1} + t_i$ for $i \in [n]$ (finishing times of greedy)

Easy to see that optimal strategy has no *idle time*.

Earliest Deadline First analysis

- We are assuming that $d_1 \leq d_2 \leq \cdots \leq d_n$
- let $f_0 := 0$ and $f_i := f_{i-1} + t_i$ for $i \in [n]$ (finishing times of greedy)
- Let $\Pi := (i_1, \ldots, i_n)$ be an optimal scheduling

Earliest Deadline First analysis

- We are assuming that $d_1 \leq d_2 \leq \cdots \leq d_n$
- let $f_0 := 0$ and $f_i := f_{i-1} + t_i$ for $i \in [n]$ (finishing times of greedy)
- Let $\Pi := (i_1, \ldots, i_n)$ be an optimal scheduling
- If $\Pi \neq (1, \ldots, n)$ (i.e. different from greedy), then Π has an inversion

Earliest Deadline First analysis

- We are assuming that $d_1 \leq d_2 \leq \cdots \leq d_n$
- let $f_0 := 0$ and $f_i := f_{i-1} + t_i$ for $i \in [n]$ (finishing times of greedy)
- Let $\Pi := (i_1, \ldots, i_n)$ be an optimal scheduling
- If $\Pi \neq (1, \ldots, n)$ (i.e. different from greedy), then Π has an inversion
- $i_k > i_{k+1} \Rightarrow d_{i_k} \geq d_{i_{k+1}}$ so after swapping/exchanging (say solution becomes Π'):

$$\begin{aligned}
L(\Pi) - L(\Pi') &= \ell_{i_{k+1}}(\Pi) - \max\{\ell_{i_{k+1}}(\Pi'), \ell_{i_k}(\Pi')\} \\
&= \max\{0, g_{k+1} - d_{i_{k+1}}\} \\
&\quad - \max(\max\{0, g_{k-1} + t_{i_{k+1}} - d_{i_{k+1}}\}, \max\{0, g_{k+1} - d_{i_k}\})
\end{aligned}$$

where $g_k := \sum_{j=1}^{k} t_{i_j}$.

Earliest Deadline First analysis

- We are assuming that $d_1 \leq d_2 \leq \cdots \leq d_n$
- let $f_0 := 0$ and $f_i := f_{i-1} + t_i$ for $i \in [n]$ (finishing times of greedy)
- Let $\Pi := (i_1, \ldots, i_n)$ be an optimal scheduling
- If $\Pi \neq (1, \ldots, n)$ (i.e. different from greedy), then Π has an inversion
- $i_k > i_{k+1} \Rightarrow d_{i_k} \geq d_{i_{k+1}}$ so after swapping/exchanging (say solution becomes Π'):

$$\begin{aligned} L(\Pi) - L(\Pi') &= \ell_{i_{k+1}}(\Pi) - \max\{\ell_{i_{k+1}}(\Pi'), \ell_{i_k}(\Pi')\} \\ &= \max\{0, g_{k+1} - d_{i_{k+1}}\} \\ &\quad - \max(\max\{0, g_{k-1} + t_{i_{k+1}} - d_{i_{k+1}}\}, \max\{0, g_{k+1} - d_{i_k}\}) \end{aligned}$$

  where $g_k := \sum_{j=1}^{k} t_{i_j}$.
- Since

$$g_{k+1} - d_{i_{k+1}} \geq g_{k-1} + t_{i_{k+1}} - d_{i_{k+1}}$$

  and

$$g_{k+1} - d_{i_{k+1}} \geq g_{k+1} - d_{i_k}$$

  we are done

Acknowledgement

- Knapsack based on [CLRS 2009, Chapter 16]
- Scheduling problem based on [Kleinberg Tardos 2006, Chapter 4.2]

References I

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford. (2009)

Introduction to Algorithms, third edition.

MIT Press

Kleinberg, Jon and Tardos, Eva (2006)

Algorithm Design.

Addison Wesley