Algorithm Theory, Tutorial 7

Johannes Kalmbach

University of Freiburg

johannes.kalmbach@gmail.com

February 2020

General Hints

- Contact tutor (`johannes.kalmbach@gmail.com`) for questions concerning corrections etc.
- Contact forum (`daphne.informatik.uni-freiburg.de`) for everything else
- Suggestion: Submit in groups of two (better for understandable algorithms)
- Submit readable solutions (LaTeX as pdf, CLEAN handwriting (+ good scan if necessary))
- Spend enough time on exercise sheets and writeup (you and I have to understand your submission).

Exercise 1, Load Balancing

- Given m machines
- Given n jobs, each job i has processing time t_i
- Assign each job to a machine to minimize **makespan**
- Makespan: largest total processing time of any machine.
- *modified greedy* algorithm: Go through jobs by decreasing length, assign to machine with smallest current load [1]
- Modified Greedy has approximation ratio of $3/2$.
- In this exercise, we want to prove that the algorithm has an even better approximation ratio.

[1] If there are machines with the same load, Greedy chooses the one with the smallest ID.

1a

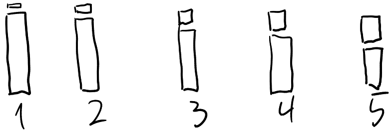
- Assume we have n jobs with lengths $t_1 \geq t_2 \geq \cdots \geq t_n$.
- let i be a machine with load T ← makespan of greedy
- let $\hat{n}$ be the last job that is scheduled on machine i.
- Shortly argue why it is sufficient to ignore jobs $\hat{n}+1, \ldots, n$ and instead prove the desired ratio between greedy and optimal for jobs $1, \ldots, \hat{n}$.
-
- After scheduling job $\hat{n}$ greedy already has makespan T (equal to the final makespan with all jobs).
- So ignoring the jobs after $\hat{n}$ doesn't make the task easier for greedy.
- The optimal solution however might be better if we leave out these jobs.
- Thus the approximation ratio on the modified problem is $\geq$ than on the original problem

Exercise 2b

- Assume we have n jobs with lengths $t_1 \geq t_2 \geq \cdots \geq t_n$.
- let i be a machine with load T
- let $\hat{n}$ be the last job that is scheduled on machine i.
- Show that if an optimal solution for jobs $1, \ldots, \hat{n}$ assigns at most two jobs to each machine, the algorithm computes an optimal solution.
  Hint: Think of a "canonical" way to assign at most two jobs to each machine and show that $T \leq T_{canonical} \leq T_{opt}$.

The canonical solution

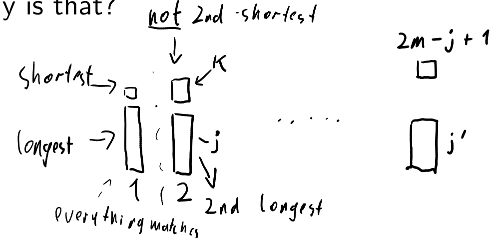
- We know that there are $\leq 2m$ jobs.
- Fill up with empty jobs, s.t. there are exactly $2m$ jobs (does not change the problem, optimal solution with max 2 jobs/machine still exists).
- Put the longest and the shortest job on machine 1.
- Put the second longest and second shortest job on machine 2.
- Put the i-th and the $2m - i + 1$-th job on machine i
- Claim: This is an optimal solution



Proof: Canonical is optimal

- Proof by exchange argument: Transform optimal into canonical solution without making it worse.
- Assume, j is the first index s.t. job j is not paired with job $2m - j + 1$ in the optimal solution.
- Job j is then paired with another job k and job $2m - 1 + 1$ is paired with another job j'.
- We can swap jobs $2m - 1 + 1$ and job k without increasing the makespan. Then we still have an optimal solution but one more index matches with the canonical solution.
- Why is that?

example:
$j = 2$



$t_j + t_k \geq t_{j'} + t_{2m-j+1}$

$t_j \geq t_{j'}$

$t_k \geq t_{2m-j+1}$

After swap:

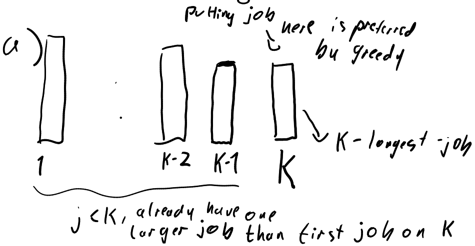
$t_j + t_{2m-j+1} \leq t_j + t_k$

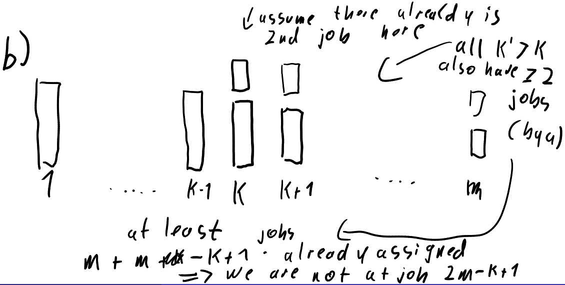
$t_{j'} + t_k \leq t_j + t_k$ $\Rightarrow$ swap is ok

Greedy is optimal

- Our canonical solution is optimal, thus we have to show that $T_{Greedy} \leq T_{Canonical}$
- Observation: for the first m jobs, greedy and canonical do the same
- Claim: While performing greedy, if machine k has only one job, then all machines $j < k$ also have only one job. Why? a)
- Thus: When greedy assigns job $2m - k + 1$, then machine k has only one job. Why? b)

  k-shortest-job
- Thus, for each job greedy can always perform the same choice as the canonical solution, and thus never gets worse than it.





- Greed-y can Schedule each job $\geq 2m-K+1$ with its Canonical partner $\Rightarrow$ greedy never exceeds Makespan of Canonical = Optimal solution

Exercise 1c

- Assume we have n jobs with lengths $t_1 \geq t_2 \geq \cdots \geq t_n$.
- let i be a machine with load T
- let $\hat{n}$ be the last job that is scheduled on machine i.
- Show that therefore, either $t_{\hat{n}} \leq T_{opt}/3$ or the greedy algorithm computes an optimal solution.
  shortest job
- If there is an optimal solution with ≤ 2 jobs per machine, then greedy is optimal.
- Else, there is a machine in the optimal solution with at least 3 jobs.
- $\hat{n}$ is the shortest job, and thus each of those jobs is $\geq t_{\hat{n}}$
- Thus there is machine that has a load of at least $3 \cdot t_{\hat{n}} \leq T_{opt}$, and thus $t_{\hat{n}} \leq T_{opt}/3$

  job/machine min. job duration

Exercise 1d

- Assume we have n jobs with lengths $t_1 \geq t_2 \geq \dots \geq t_n$.
- let i be a machine with load T
- let $\hat{n}$ be the last job that is scheduled on machine i.
- Conclude that the algorithm has an approximation ratio of at most $4/3$.
- We only have to look at the case, where the optimal solution has at least three jobs for at least one machine, otherwise we are optimal.
- Greedy assigns job $\hat{n}$ (Last job) to the machine with minimal load. And this machine has maximum load afterwards
- Minimal load before adding $\hat{n} \leq$ Average load before adding $\hat{n}$ $\leq T_{opt}$
- Since $t_{\hat{n}} \leq T_{opt}/3$ we have

$$T_{greedy} \leq 4/3\, T_{opt} \quad = \underbrace{T_{opt}}_{\substack{\text{upper bound}\\ \text{load of machine } i\\ \text{before adding } \hat{n}}} + \underbrace{\frac{T_{opt}}{3}}_{\substack{\text{upper bound}\\ \text{for } t_{\hat{n}}}}$$

- Show that the 4/3 bound is tight, i.e., there is a sequence of instances for which the ratio between Greedy and OPT converges to 4/3.
  Hint: Consider $2m+1$ jobs for m machines, three jobs with processing time m and two jobs with processing times $m+1, m+2, \ldots, 2m-1$ each.

- 1st job | 2nd job // so far each machine has load $3m-1$, still have to add 1 job of length m (to any of the machines)

$\Rightarrow$ greedy value is $4m-1$

m machines

$$\left.\begin{array}{l} 2m-1 + m \\ 2m-1 + m \\ 2m-2 + m+1 \\ 2m-2 + m+1 \\ \ldots \\ \ldots \\ \ldots \\ 1.5m \quad 1.5m-1 \\ 1.5m \quad 1.5m-1 \end{array}\right.$$

Optimal:

$$\begin{array}{l} \overset{1}{m} + \overset{2}{m} + \overset{3}{m} \\ 2m-1+m+1 \\ 2m-1+m+1 \\ 2m-2+m+2 \\ \ldots \\ \ldots \end{array}$$

$\Rightarrow$ Optimal solution is $3m$

$\Rightarrow$ Approx. ratio $\frac{4m-1}{3m} \xrightarrow{m \to \infty} \frac{4}{3}$

Consider the following variation of the knapsack problem: Given items $1, \dots, n$ where each item i has a positive integer *weight* $w_i \in \mathbb{N}$ and a positive *value* $v_i > 0$ and **two** knapsacks of capacities W_1 and W_2, we want to pack the items into the knapsacks such that

- for $j \in \{1, 2\}$, the *total weight* of the items in knapsack j is at most W_j.
- The *total value* of the items that are packed in either knapsack is maximized.

2a

- Prove that this problem is not equivalent to the standard knapsack problem with one knapsack of capacity $W_1 + W_2$ by showing that the total value that can be packed into one knapsack of capacity $W_1 + W_2$ can be *arbitrarily* larger than the total value that can be packed into two knapsacks of capacities W_1 and W_2.

$W_1 = 1$

$W_2 = 1$

2 knapsacks solution value = 0

$W_1 + W_2$ solution value = x

1 item with weight 2, value x

arbitrarily big

2b

- Assume that $W_1 \geq W_2$. A simple strategy would be to first compute an optimal solution for a knapsack of capacity W_1 and afterwards, with the remaining elements, an optimal solution for a knapsack of capacity W_2. Show that this algorithm always computes at least a 2-approximation for the problem.
- Let OPT_{W1} be the content of $W1$ in the optimal solution of the overall problem, OPT_{W2} similarly.
- let $OPT_{max} := max(OPT_{W1}, OPT_{W2})$
- We know that $OPT_{max} \geq 0.5(OPT_{W1} + OPT_{W2})$ ✓ max ≥ average
- Also we know that OPT_{max} fits into W1 (because it is the bigger knapsack). (feasible for Knapsack on W_1
- Let $OPTSINGLE_{W1}$ be an optimal solution on the "standard knapsack" on $W1$.
- We know that $OPTSINGLE_{W1} \geq OPT_{max} \geq 0.5(OPT_{W1} + OPT_{W2})$
- So even calculating the traditional knapsack on W1 and ignoring W2 is a 2-approximation, so this also holds if we put some of the rest into W2

Exercise 3

- Show that taking all nodes is a 2-approximation algorithm for the vertex cover problem in regular graphs (graphs where all nodes have the same degree) $n :=$ # vertices / nodes
- A r-regular graph has $\frac{n \cdot r}{2}$ edges ~~nodes~~
- each of the n nodes covers at most r (additional) edges
- so to cover all edges, we need at least $\frac{n}{2}$ nodes (every vertex cover on regular graphs has size $\geq \frac{n}{2}$
- Taking all n nodes is a 2-Approximation