Algorithm Theory

Chapter 8
Approximation Algorithms

Part I:
Greedy Load Balancing

Fabian Kuhn

Approximation Algorithms

- Optimization appears everywhere in computer science
- We have seen several examples, e.g.:
  - scheduling jobs
  - traveling salesperson
  - maximum flow, maximum matching
  - minimum spanning tree
  - …
- Many discrete optimization problems are NP-hard
- They are however still important and we need to solve them
- As algorithm designers, we prefer algorithms that produce solutions which are provably good, even if we can't compute an optimal solution.

Approximation Algorithms: Examples

We have already seen two approximation algorithms

- **Metric TSP:** If distances are positive and satisfy the triangle inequality, the greedy tour is only by a log-factor longer than an optimal tour

- **Maximum Matching :** A maximal matching gives a solution that is within a factor of 2 of a maximum matching.

Approximation Ratio

An approximation algorithm is an algorithm that computes a solution for an optimization with an objective value that is provably within a bounded factor of the optimal objective value.

Formally:

- $\mathrm{OPT}(I) \geq 0$: optimal objective value
  $\mathrm{ALG}(I) \geq 0$: objective value achieved by the algorithm

- **Approximation Ratio α:**

  Minimization: $\alpha := \max\limits_{\text{input instances } I} \dfrac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)}$

  Maximization: $\alpha := \min\limits_{\text{input instances } I} \dfrac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)}$

For maximization, sometimes also $\alpha := \max\limits_{\text{input inst. } I} \dfrac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$

Example: Load Balancing

We are given:

- m machines $M_1, \dots, M_m$
- n jobs, processing time of job i is $t_i > 0$

Goal:

- Assign each job to a machine such that the makespan is minimized

  makespan: largest total processing time of any machine

The above load balancing problem is NP-hard and we therefore want to get a good approximation for the problem.

Greedy Algorithm

There is a simple greedy algorithm:

- Go through the jobs in an arbitrary order
- When considering job i, assign the job to the machine that currently has the smallest load.

Example: 3 machines, 12 jobs

3	4	2	3	1	6	4	4	3	2	1	5

Greedy Assignment:

M_1:	3	1	6	1	5
M_2:	4	4	3		
M_3:	2	3	4	2	

$\text{makespan} = 16$

Optimal Assignment:

M_1:	3	4	2	3	1
M_2:	6	4	3		
M_3:	4	2	1	5	

$\text{makespan} = 13$

Greedy Analysis

- We will show that greedy gives a 2-approximation

- To show this, we need to compare the solution of greedy with an optimal solution (that we can't compute)

- Lower bound on the optimal makespan T^*:

$$T^* \geq \frac{1}{m} \cdot \sum_{i=1}^{n} t_i$$

- Lower bound can be far from T^*:
  - m machines, m jobs of size 1, 1 job of size m

$$T^* = m, \qquad \frac{1}{m} \cdot \sum_{i=1}^{n} t_i = 2$$

Greedy Analysis

- We will show that greedy gives a 2-approximation
- To show this, we need to compare the solution of greedy with an optimal solution (that we can't compute)
- Lower bound on the optimal makespan T^*:

$$T^* \geq \frac{1}{m} \cdot \sum_{i=1}^{n} t_i$$

- Second lower bound on optimal makespan T^*:

$$T^* \geq \max_{1 \leq i \leq n} t_i$$

Greedy Analysis

Theorem: The greedy algorithm has approximation ratio < 2, i.e., for the makespan T of the greedy solution, we have $T < 2T^*$.

Proof:

- For machine k, let T_k be the time used by machine k
- Consider some machine M_i for which $T_i = T$
- Assume that job j is the last one assigned to M_i:

$$\boldsymbol{M_i}\text{:}\quad \underbrace{\boxed{\quad T - t_j \quad}}_{\text{minimum load when job } j \text{ is scheduled}}\boxed{\quad t_j \quad}$$

- When job j is assigned, M_i has the minimum load

$$\forall k \in \{1, \dots, m\} : T_k \geq T - t_j \implies \underbrace{\sum_{x=1}^{n} t_x > m \cdot (T - t_j)}_{\text{avg. load} > T - t_j}$$

Greedy Analysis

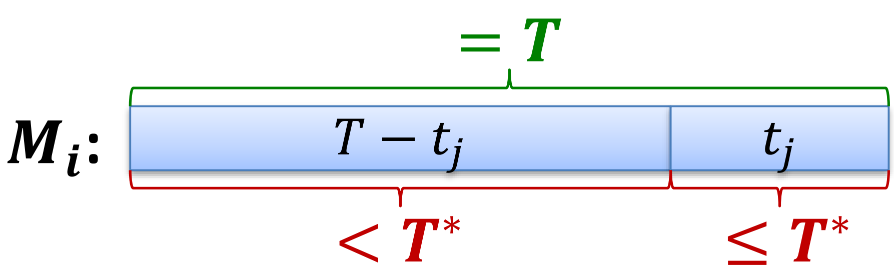
Theorem: The greedy algorithm has approximation ratio < 2, i.e., for the makespan T of the greedy solution, we have $T < 2T^*$.

Proof:

- For all machines M_k, load $T_k \geq T - t_j$:

$$T^* \geq \frac{1}{m} \cdot \sum_{x=1}^{n} t_x > T - t_j$$

- In greedy solution, machine M_i has maximum load T

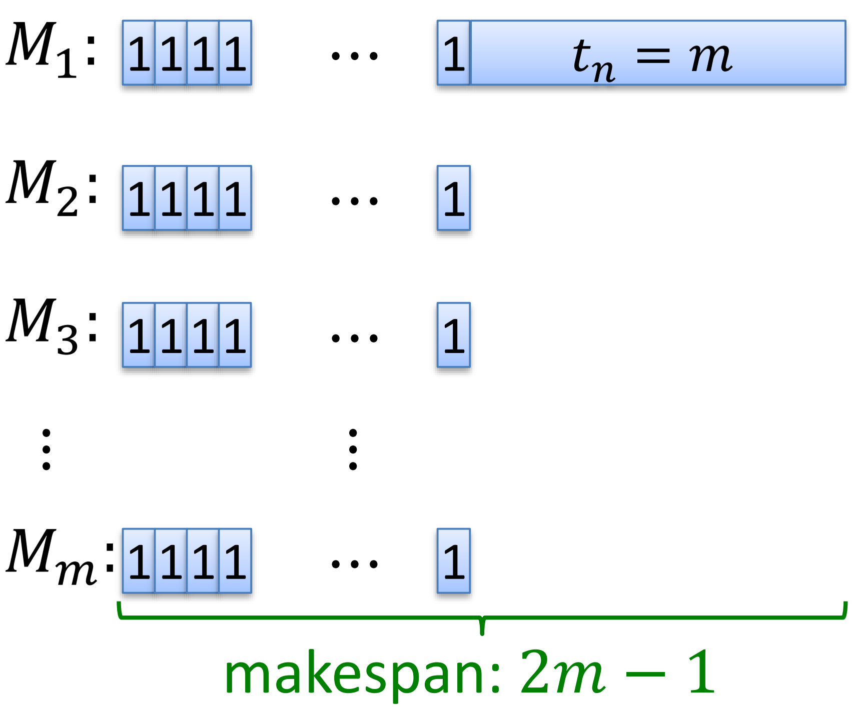


Can We Do Better?

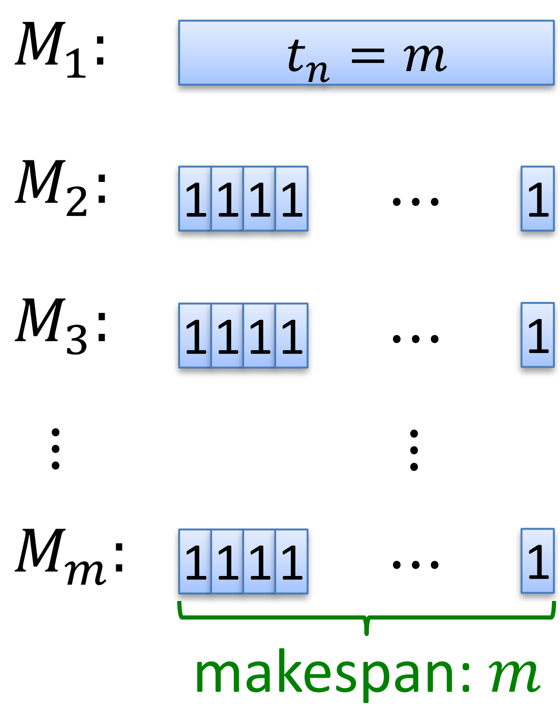
The analysis of the greedy algorithm is almost tight:

- Example with $n = m(m-1)+1$ jobs
- Jobs $1, \ldots, n-1 = m(m-1)$ have $t_i = 1$, job n has $t_n = m$

Greedy Assignment:



Optimal Assignment:



Improving Greedy

Bad case for the greedy algorithm:
One large job in the end can destroy everything

Idea: assign large jobs first

Modified Greedy Algorithm:

1. Sort jobs by decreasing length s.t. $t_1 \geq t_2 \geq \cdots \geq t_n$
2. Apply the greedy algorithm as before (in the sorted order)

Lemma: If $n > m$: $T^* \geq t_m + t_{m+1} \geq 2t_{m+1}$

Proof:

- Two of the first $m + 1$ jobs need to be assigned to the same machine
- Jobs m and $m + 1$ are the shortest of these jobs

Analysis of the Modified Greedy Alg.

Theorem: The modified algorithm has approximation ratio $< {}^3/_2$.

Proof:

- We show that $T < {}^3/_2 \cdot T^*$
- As before, we consider the machine M_i with $T_i = T$
- Job j (of length t_j) is the last one assigned to machine M_i
- If j is the only job on M_i, we have $T = T^*$
- Otherwise, we have $j \geq m + 1$
  - The first m jobs are assigned to m distinct machines

$$\boldsymbol{M_i}: \overbrace{\underbrace{\boxed{T - t_j}}_{< T^*}\underbrace{\boxed{t_j}}_{\leq t_{m+1} \leq T^*/2}}^{= T}$$

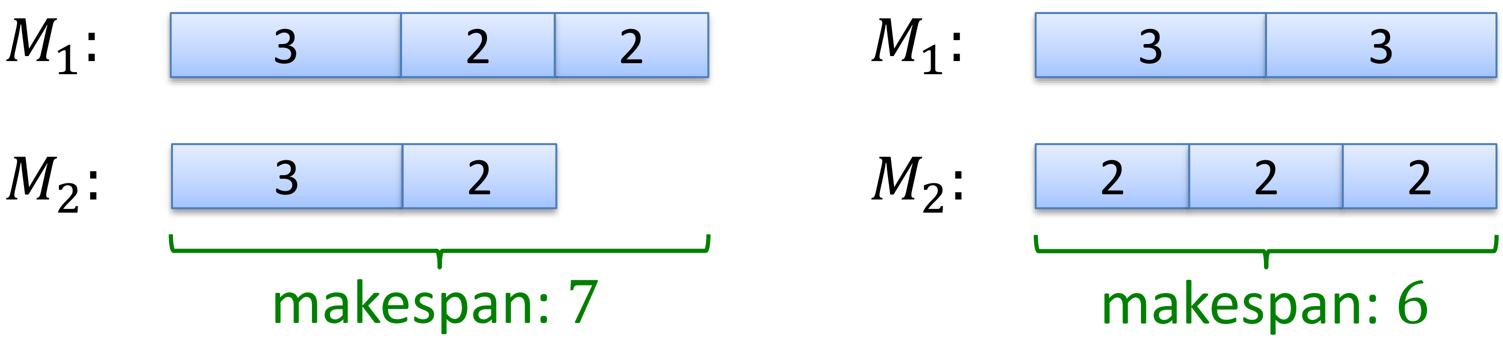
Analysis of the Modified Greedy Alg.

Theorem: The modified algorithm has approximation ratio $\geq {}^{7}/_{6}$.

Proof:

- Example with 5 jobs and 2 machines: $t_1, t_2 = 3,\ t_3, t_4, t_5 = 2$

Greedy Assignment: **Optimal Assignment:**



- **Remark:** Both bounds are not tight
  - Modified greedy algorithm has approximation ratio $< 4/3$
  - One can construct an example, where the approximation is arbitrarily close to $4/3$