Lecture: Approximation Algorithms

Jannik Matuschke

TUM

November 7, 2018

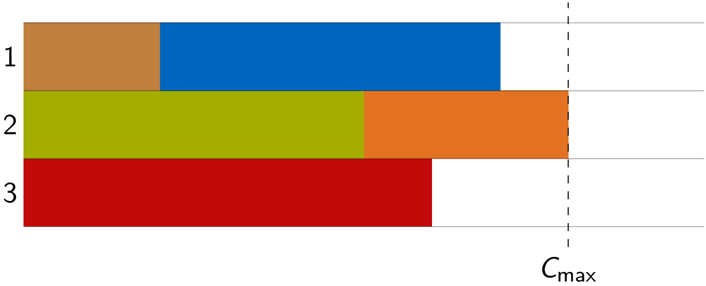
Dynamic Programming

Example II: Scheduling on Identical Parallel Machines

Scheduling on Parallel Machines

Input: m identical machines,
n jobs with processing times $p_1, \dots, p_n$

Task: assign each job $j \in [n]$ to a machine $\sigma(j) \in [m]$
minimizing $C_{\max} := \max_{i \in [m]} \sum_{j\,:\,\sigma(j)=i} p_j$



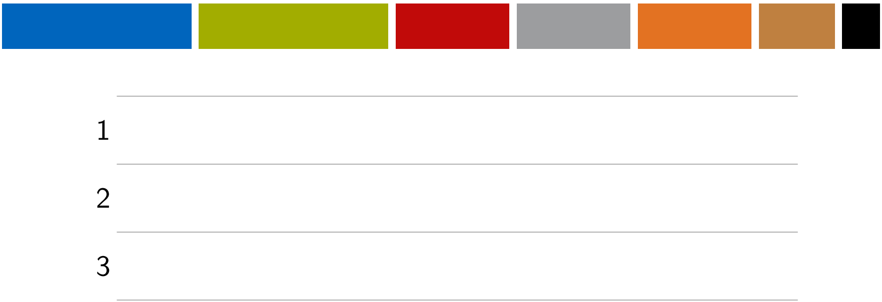
An approximation scheme

Algorithm $A(k)$

j' long: $p_{j'} \geq \frac{1}{km} \sum_{j \in [n]} p_j$

1. Compute optimal schedule for long jobs. (complete enumeration)
2. For each short job j

    Assign j to machine i with lowest load.



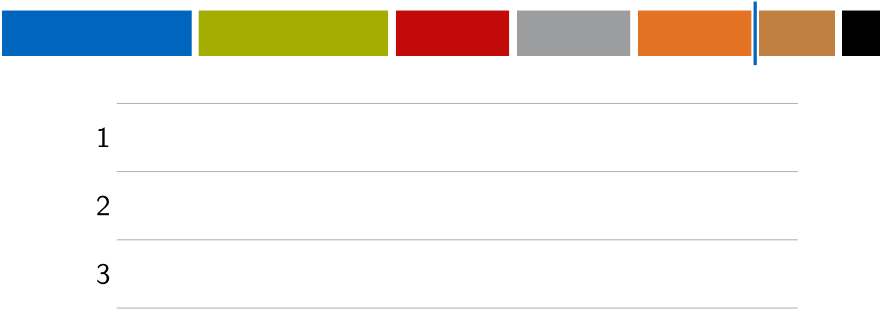
An approximation scheme

Algorithm $A(k)$

j' long: $p_{j'} \geq \frac{1}{km} \sum_{j \in [n]} p_j$

1. Compute optimal schedule for long jobs. (complete enumeration)
2. For each short job j

   Assign j to machine i with lowest load.



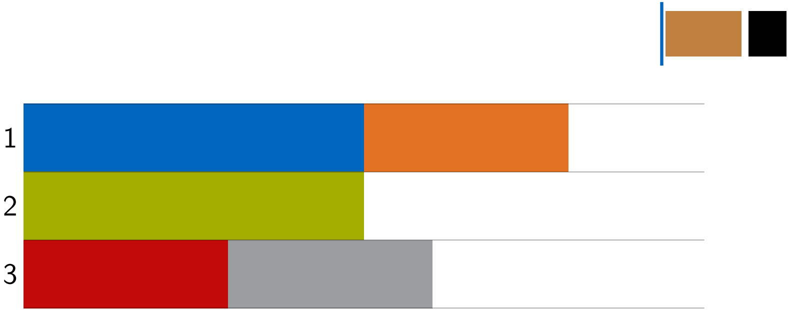
An approximation scheme

Algorithm $A(k)$

j' long: $p_{j'} \geq \frac{1}{km} \sum_{j \in [n]} p_j$

1. Compute optimal schedule for long jobs. (complete enumeration)
2. For each short job j

   Assign j to machine i with lowest load.

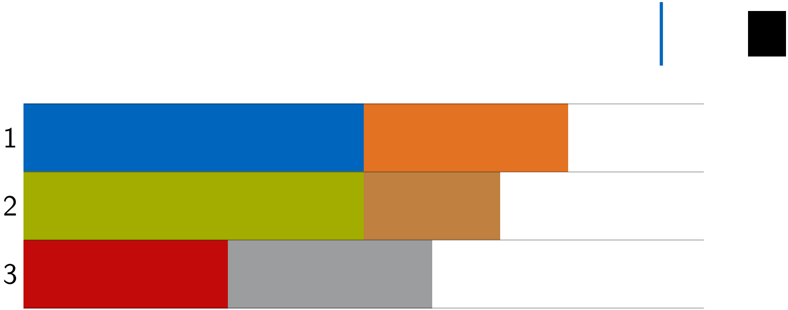


An approximation scheme

Algorithm $A(k)$ $\qquad$ j' long: $p_{j'} \geq \frac{1}{km} \sum_{j \in [n]} p_j$

1. Compute optimal schedule for long jobs. (complete enumeration)
2. For each short job j

   Assign j to machine i with lowest load.



An approximation scheme

Algorithm $A(k)$

j' long: $p_{j'} \geq \frac{1}{km} \sum_{j \in [n]} p_j$

1. Compute optimal schedule for long jobs. (complete enumeration)
2. For each short job j

   Assign j to machine i with lowest load.

An approximation scheme

Algorithm $A(k)$ j' long: $p_{j'} \geq \frac{1}{km} \sum_{j \in [n]} p_j$

1. Compute optimal schedule for long jobs. (complete enumeration)
2. For each short job j

   Assign j to machine i with lowest load.

Theorem 5.5

$A(k)$ is a $(1 + 1/k)$-approximation with running time $O(m^{km} + n)$.

An approximation scheme

Algorithm $A'(k, T)$

j long: $p_j \geq \frac{T}{k}$

1. Schedule long jobs within $(1 + 1/k)T$ (or find out that $T < \mathrm{OPT}$ and stop).
2. For each short job j

   Assign j to machine i with lowest load.

An approximation scheme

Algorithm $A'(k, T)$

j long: $p_j \geq \frac{T}{k}$

1. Schedule long jobs within $(1 + 1/k)T$
   (or find out that $T < \mathrm{OPT}$ and stop).
2. For each short job j

   Assign j to machine i with lowest load.

Theorem 5.6

If $A'(k, T)$ computes a schedule, then it has makespan $(1 + 1/k)T$.

An approximation scheme

Algorithm $A'(k, T)$ j long: $p_j \geq \frac{T}{k}$

1. Schedule long jobs within $(1 + 1/k)T$ $\rightarrow$ B(k, T)
   (or find out that $T <$ OPT and stop).
2. For each short job j
   Assign j to machine i with lowest load.

Theorem 5.6

If $A'(k, T)$ computes a schedule, then it has makespan $(1 + 1/k)T$.

Subroutine for long jobs

Algorithm $B(k, T)$ only long jobs: $p_j \geq \frac{T}{k}$

1. $p'_j := \left\lfloor \frac{k^2 p_j}{T} \right\rfloor \cdot \frac{T}{k^2}$ for each j
2. Compute $m' :=$ min number of machines needed to schedule rounded long jobs within makespan T.
3. If $m' \leq m$ then return corresponding schedule.
4. Otherwise return "failed".

Subroutine for long jobs

Algorithm $B(k, T)$ only long jobs: $p_j \geq \frac{T}{k}$

1. $p'_j := \left\lfloor \frac{k^2 p_j}{T} \right\rfloor \cdot \frac{T}{k^2}$ for each j
2. Compute $m' :=$ min number of machines needed to schedule rounded long jobs within makespan T. $\rightarrow$ DP
3. If $m' \leq m$ then return corresponding schedule.
4. Otherwise return "failed".

Lemma 5.7

If $B(k, T)$ computes a schedule, then it has makespan $(1 + 1/k)T$.
If $B(k, T)$ returns "failed", then $\mathrm{OPT} > T$.

Subroutine for long jobs

Algorithm $B(k, T)$ only long jobs: $p_j \geq \frac{T}{k}$

1. $p'_j := \left\lfloor \frac{k^2 p_j}{T} \right\rfloor \cdot \frac{T}{k^2}$ for each j
2. Compute $m' :=$ min number of machines needed to schedule rounded long jobs within makespan T. $\rightarrow$ DP
3. If $m' \leq m$ then return corresponding schedule.
4. Otherwise return "failed".

Lemma 5.7

If $B(k, T)$ computes a schedule, then it has makespan $(1 + 1/k)T$.
If $B(k, T)$ returns "failed", then $\mathrm{OPT} > T$.

Lemma 5.8

$B(k, T)$ runs in time $O(n^{k^2}(k+1)^{k^2})$.

Dynamic program: Idea

Idea:

- only k^2 different job types $(p'_j \in \{i \cdot \frac{T}{k^2} \,:\, i \in [k^2]\})$

Dynamic program: Idea

Idea:

- only k^2 different job types $\left(p'_j \in \{i \cdot \frac{T}{k^2} \ : \ i \in [k^2]\}\right)$
- n_i: number of jobs of type i

Dynamic program: Idea

Idea:

- only k^2 different job types ($p'_j \in \{i \cdot \frac{T}{k^2} : i \in [k^2]\}$)
- n_i: number of jobs of type i
- $M(n_1, \ldots, n_{k^2}) := \#$machines needed to schedule instance defined by $n_1, \ldots, n_{k^2}$

Dynamic program: Idea

Idea:

- only k^2 different job types ($p'_j \in \{i \cdot \frac{T}{k^2} \ : \ i \in [k^2]\}$)
- n_i: number of jobs of type i
- $M(n_1, \ldots, n_{k^2}) :=$ #machines needed to schedule instance defined by $n_1, \ldots, n_{k^2}$

$$\#\text{instances} \leq n^{k^2}$$

Dynamic program: Idea

Idea:

- only k^2 different job types ($p'_j \in \{i \cdot \frac{T}{k^2} \ : \ i \in [k^2]\}$)
- n_i: number of jobs of type i
- $M(n_1, \dots, n_{k^2}) :=$ #machines needed to schedule instance defined by $n_1, \dots, n_{k^2}$
- $M(n_1, \dots, n_{k^2}) = 1 + \min_{s \in \mathcal{C}} M(n_1 - s_1, \dots, n_{k^2} - s_{k^2})$
- $\mathcal{C} :=$ configurations (schedule s_i jobs of type i on the machine)

$$\#\text{instances} \leq n^{k^2}$$

Dynamic program: Idea

Idea:

- only k^2 different job types ($p'_j \in \{i \cdot \frac{T}{k^2} \,:\, i \in [k^2]\}$)
- n_i: number of jobs of type i
- $M(n_1, \ldots, n_{k^2}) :=$ #machines needed to schedule instance defined by $n_1, \ldots, n_{k^2}$
- $M(n_1, \ldots, n_{k^2}) = 1 + \min_{s \in \mathcal{C}} M(n_1 - s_1, \ldots, n_{k^2} - s_{k^2})$
- $\mathcal{C} :=$ configurations (schedule s_i jobs of type i on the machine)

$$\#\text{instances} \leq n^{k^2} \quad \text{and} \quad |\mathcal{C}| \leq (k+1)^{k^2}$$

Dynamic program: Idea

Idea:

- only k^2 different job types ($p'_j \in \{i \cdot \frac{T}{k^2} \,:\, i \in [k^2]\}$)
- n_i: number of jobs of type i
- $M(n_1, \dots, n_{k^2}) :=$ #machines needed to schedule instance defined by $n_1, \dots, n_{k^2}$
- $M(n_1, \dots, n_{k^2}) = 1 + \min_{s \in \mathcal{C}} M(n_1 - s_1, \dots, n_{k^2} - s_{k^2})$
- $\mathcal{C} :=$ configurations (schedule s_i jobs of type i on the machine)

$$\#\text{instances} \leq n^{k^2} \quad \text{and} \quad |\mathcal{C}| \leq (k+1)^{k^2}$$

$$\text{dynamic program: } n^{k^2}(k+1)^{k^2}$$

Dynamic program: Pseudocode

Algorithm DP

1. $M(0, \ldots, 0) := 0,\ Q := \{(0, \ldots, 0)\}$
2. while $(Q \neq \emptyset)$

    Let $q \in Q$ such that $M(q)$ is minimum.
    for each $s \in \mathcal{C}$ with $q_i + s_i \leq n$ for all i
        if $M(q) + 1 < M(q + s)$ then
            $M(q + s) := M(q) + 1$
            $Q := Q \cup \{q + s\}$
        end if
    end for
    $Q := Q \setminus \{q\}$

3. Return $M(n_1, \ldots, n_{k^2})$.

An approximation scheme

Algorithm $A'(k, T)$ j long: $p_j \geq \frac{T}{k}$

1. Schedule long jobs within $(1 + 1/k)T$ $\rightarrow$ B(k, T)
   (or find out that $T <$ OPT and stop).
2. For each short job j
   Assign j to machine i with lowest load.

Theorem 5.9

If $A'(k, T)$ computes a schedule, then it has makespan $(1 + 1/k)T$.
$A'(k, T)$ runs in time $O(n^{k^2}(k + 1)^{k^2})$.

An approximation scheme

Algorithm $A'(k, T)$ $\qquad$ j long: $p_j \geq \frac{T}{k}$

1. Schedule long jobs within $(1 + 1/k)T$ $\rightarrow$ B(k, T)
   (or find out that $T <$ OPT and stop).
2. For each short job j
   Assign j to machine i with lowest load.

Theorem 5.9

If $A'(k, T)$ computes a schedule, then it has makespan $(1 + 1/k)T$.
$A'(k, T)$ runs in time $O(n^{k^2}(k+1)^{k^2})$.

$A'(k, T)$ is a $(1 + 1/k)$-relaxed decision procedure.

Can be turned into a $(1 + 1/k)$-approximation algorithm. (Exercise)

PTAS: choose $k := \lceil 1/\varepsilon \rceil$

Why no FPTAS?

Theorem 5.10

There is a polynomial function q such that $P||C_{\max}$ is NP-hard even when restricted to instances with $p_j \leq q(n)$ for all j.

"$P||C_{\max}$ is strongly NP-hard."

Why no FPTAS?

Theorem 5.10

There is a polynomial function q such that $P||C_{\max}$ is NP-hard even when restricted to instances with $p_j \leq q(n)$ for all j.

"$P||C_{\max}$ is strongly NP-hard."

Corollary 5.11

There is no FPTAS for $P||C_{\max}$, unless $P = NP$.

Why no FPTAS?

Theorem 5.10

There is a polynomial function q such that $P||C_{\max}$ is NP-hard even when restricted to instances with $p_j \leq q(n)$ for all j.

"$P||C_{\max}$ is strongly NP-hard."

Corollary 5.11

There is no FPTAS for $P||C_{\max}$, unless $P = NP$.

Proof. Consider instance with $p_j \leq q(n)$ for all $j \in [n]$. $Q := nq(n)$
Run FPTAS with $\varepsilon := 1/(Q+1)$. $\mathrm{OPT} \leq Q$

Why no FPTAS?

Theorem 5.10

There is a polynomial function q such that $P||C_{\max}$ is NP-hard even when restricted to instances with $p_j \leq q(n)$ for all j.

"$P||C_{\max}$ is strongly NP-hard."

Corollary 5.11

There is no FPTAS for $P||C_{\max}$, unless $P = NP$.

Proof. Consider instance with $p_j \leq q(n)$ for all $j \in [n]$. $Q := nq(n)$
Run FPTAS with $\varepsilon := 1/(Q+1)$. OPT $\leq Q$

- poly-time in input size, because $1/\varepsilon \leq nq(n) + 1$

Why no FPTAS?

Theorem 5.10

There is a polynomial function q such that $P||C_{\max}$ is NP-hard even when restricted to instances with $p_j \leq q(n)$ for all j.

"$P||C_{\max}$ is strongly NP-hard."

Corollary 5.11

There is no FPTAS for $P||C_{\max}$, unless $P = NP$.

Proof. Consider instance with $p_j \leq q(n)$ for all $j \in [n]$. $Q := nq(n)$
Run FPTAS with $\varepsilon := 1/(Q+1)$. $\mathrm{OPT} \leq Q$

- poly-time in input size, because $1/\varepsilon \leq nq(n) + 1$
- $\mathrm{ALG} \;\leq\; (1+\varepsilon)\,\mathrm{OPT} \;<\; \mathrm{OPT} + \mathrm{OPT}\,/Q \;\leq\; \mathrm{OPT} + 1$

Why no FPTAS?

Theorem 5.10

There is a polynomial function q such that $P||C_{\max}$ is NP-hard even when restricted to instances with $p_j \leq q(n)$ for all j.

"$P||C_{\max}$ is strongly NP-hard."

Corollary 5.11

There is no FPTAS for $P||C_{\max}$, unless $P = NP$.

Proof. Consider instance with $p_j \leq q(n)$ for all $j \in [n]$. $Q := nq(n)$
Run FPTAS with $\varepsilon := 1/(Q+1)$. $\mathrm{OPT} \leq Q$

- poly-time in input size, because $1/\varepsilon \leq nq(n) + 1$
- $\mathrm{ALG} \leq (1+\varepsilon)\,\mathrm{OPT} < \mathrm{OPT} + \mathrm{OPT}/Q \leq \mathrm{OPT} + 1$
  by integrality: $\mathrm{ALG} \leq \mathrm{OPT}$. $\square$