New Support Size Bounds for Integer Programming, Applied to Makespan Minimization on Uniformly Related Machines

S. Berndt[1], H. Brinkop[2], K. Jansen[2],
M. Mnich[3], and T. Stamm[3].

29 May 2024

[1] University of Lübeck, Institute for Theoretical Computer Science, Lübeck, Germany
[2] Kiel University
[3] Hamburg University of Technology, Institute for Algorithms and Complexity, Hamburg, Germany

$Q||C_{\max}$ (makespan minimization on uniform machines)

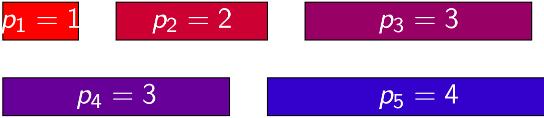
Input:

Output:

$Q||C_{\max}$ (makespan minimization on uniform machines)

Input:

- N jobs (with processing times $p_j \in \mathbb{N}$)

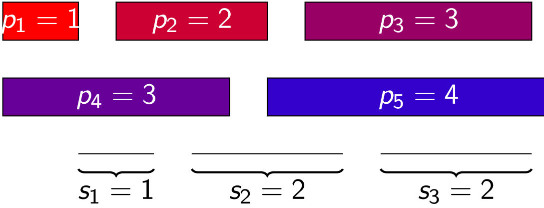


Output:

$Q||C_{\max}$ (makespan minimization on uniform machines)

Input:

- N jobs (with processing times $p_j \in \mathbb{N}$)
- M machines (with processing speeds $s_i \in \mathbb{N}$)



Output:

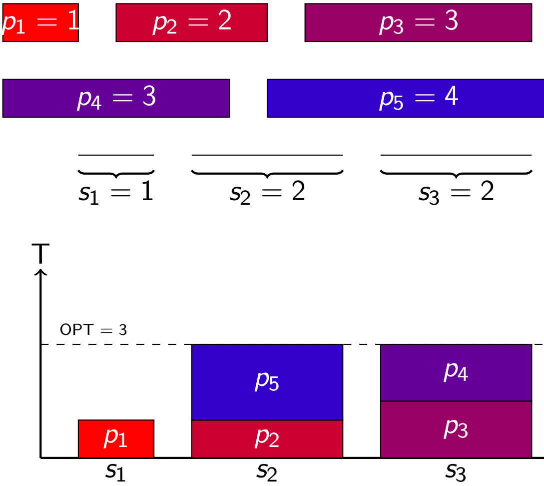
$Q||C_{\max}$ (makespan minimization on uniform machines)

Input:

- N jobs (with processing times $p_j \in \mathbb{N}$)
- M machines (with processing speeds $s_i \in \mathbb{N}$)

Output:

- Schedule σ (minimizing makespan OPT)



$Q||C_{\max}$ (makespan minimization on uniform machines)

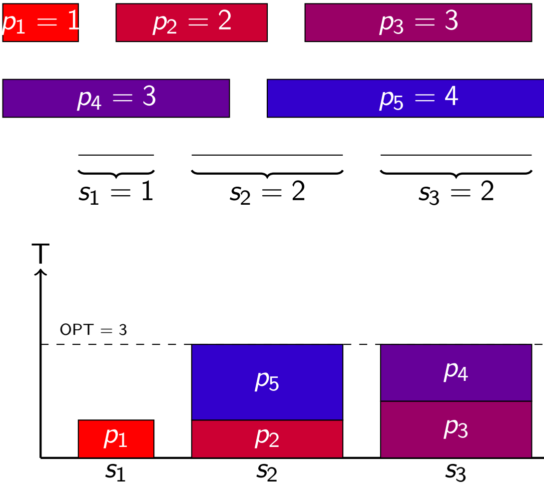
Input:

- N jobs (with processing times $p_j \in \mathbb{N}$)
- M machines (with processing speeds $s_i \in \mathbb{N}$)

Output:

- Schedule σ (minimizing makespan OPT)

$Q||C_{\max}$ is NP-hard.



$Q||C_{\max}$ (makespan minimization on uniform machines)

Input:

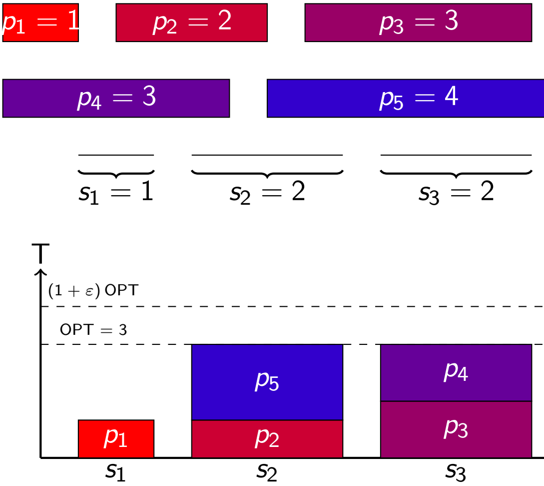
- N jobs (with processing times $p_j \in \mathbb{N}$)
- M machines (with processing speeds $s_i \in \mathbb{N}$)

Output:

- Schedule σ (minimizing makespan OPT)

$Q||C_{\max}$ is NP-hard.

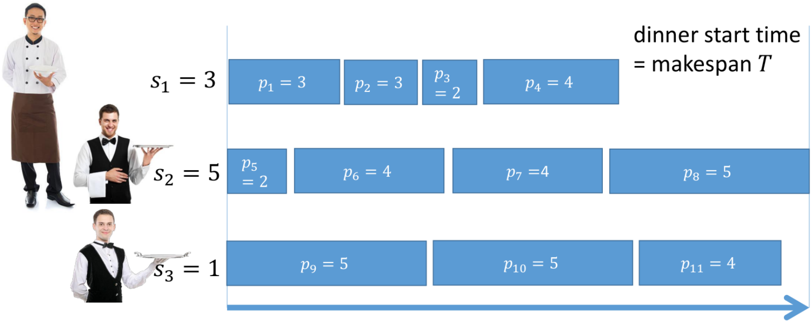
Goal: compute $(1+\varepsilon)$-approximation



Conference dinner on time

An application of $Q||C_{\max}$

m waiters serve n participants of a banquet their food as quickly as possible



Problem context

$P||C_{\max}$ (important special case with processing speeds $s_1 = \ldots = s_m$):

Jansen, Rohwedder (2018):
Few constraints ILP algorithm

$$2^{\mathcal{O}(1/\varepsilon \log^2(1/\varepsilon))} + \mathcal{O}(N)$$

Problem context

$P||C_{\max}$ (important special case with processing speeds $s_1 = \ldots = s_m$):

Jansen, Rohwedder (2018):
Few constraints ILP algorithm

$2^{\mathcal{O}(1/\varepsilon \log^2(1/\varepsilon))} + \mathcal{O}(N)$

$\rightarrow$

Berndt et al. (2021):
Small column norm constraint matrix

$2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon) \log(\log(1/\varepsilon)))} + \mathcal{O}(N)$

Problem context

$P||C_{\max}$ (important special case with processing speeds $s_1 = \ldots = s_m$):

Jansen, Rohwedder (2018):
Few constraints ILP algorithm
$2^{\mathcal{O}(1/\varepsilon \log^2(1/\varepsilon))} + \mathcal{O}(N)$

→

Berndt et al. (2021):
Small column norm constraint matrix
$2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon) \log(\log(1/\varepsilon)))} + \mathcal{O}(N)$

$Q||C_{\max}$ (arbitrary processing speeds $s_1 \leq \ldots \leq s_m$):

Jansen et al. (2016):
Few constraints MILP algorithm
$2^{\mathcal{O}(1/\varepsilon \log^4(1/\varepsilon))} + \mathcal{O}(N)$

Problem context

$P||C_{\max}$ (important special case with processing speeds $s_1 = \ldots = s_m$):

Jansen, Rohwedder (2018): Few constraints ILP algorithm $2^{\mathcal{O}(1/\varepsilon \log^2(1/\varepsilon))} + \mathcal{O}(N)$	$\rightarrow$	Berndt et al. (2021): Small column norm constraint matrix $2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon) \log(\log(1/\varepsilon)))} + \mathcal{O}(N)$

$Q||C_{\max}$ (arbitrary processing speeds $s_1 \leq \ldots \leq s_m$):

Jansen et al. (2016): Few constraints MILP algorithm $2^{\mathcal{O}(1/\varepsilon \log^4(1/\varepsilon))} + \mathcal{O}(N)$	$\rightarrow$	Our result (2023): Small column norm constraint matrix $2^{\mathcal{O}(1/\varepsilon \log^3(1/\varepsilon) \log(\log(1/\varepsilon)))} + \mathcal{O}(N)$

Run times to compute $(1+\varepsilon)$-approximate schedules for $Q||C_{\max}$

authors	year	run time
Hochbaum, Shmoys	1988	$N^{\mathcal{O}(1/\varepsilon^2 \log(1/\varepsilon))}$
Azar, Epstein	1998	$N^{\mathcal{O}(1/\varepsilon^2)}$
Jansen	2010	$2^{\mathcal{O}(1/\varepsilon^2 \log^3(1/\varepsilon))} + N^{\mathcal{O}(1)}$
Jansen, Robenek	2012	$2^{\mathcal{O}(1/\varepsilon^2 \log^3(1/\varepsilon))} + N^{\mathcal{O}(1)}$
Jansen et al.	2016	$2^{\mathcal{O}\left(1/\varepsilon \log^4(1/\varepsilon)\right)} + N^{\mathcal{O}(1)}$
Our result	2023	$2^{\mathcal{O}\left(1/\varepsilon \log^3(1/\varepsilon)\log(\log(1/\varepsilon))\right)} + \mathcal{O}(N)$

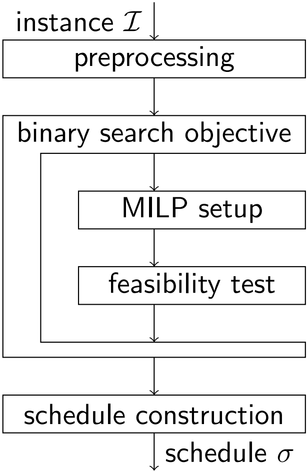
Run times to compute $(1+\varepsilon)$-approximate schedules for $Q||C_{\max}$

authors	year	run time
Hochbaum, Shmoys	1988	$N^{\mathcal{O}(1/\varepsilon^2 \log(1/\varepsilon))}$
Azar, Epstein	1998	$N^{\mathcal{O}(1/\varepsilon^2)}$
Jansen	2010	$2^{\mathcal{O}(1/\varepsilon^2 \log^3(1/\varepsilon))} + N^{\mathcal{O}(1)}$
Jansen, Robenek	2012	$2^{\mathcal{O}(1/\varepsilon^2 \log^3(1/\varepsilon))} + N^{\mathcal{O}(1)}$
Jansen et al.	2016	$2^{\mathcal{O}\left(1/\varepsilon \log^4(1/\varepsilon)\right)} + N^{\mathcal{O}(1)}$
Our result	2023	$2^{\mathcal{O}\left(1/\varepsilon \log^3(1/\varepsilon)\log(\log(1/\varepsilon))\right)} + \mathcal{O}(N)$

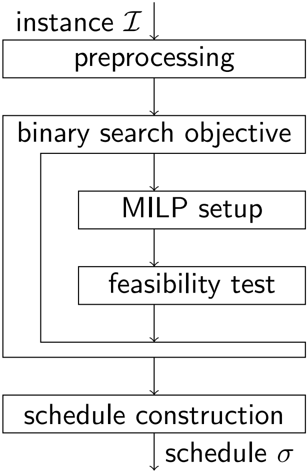
Special case $P||C_{\max}$: all machines have speed $s_1 = \ldots = s_m = 1$

Jansen, Rohwedder	2019	$2^{\mathcal{O}(1/\varepsilon \log^2(1/\varepsilon))} + \mathcal{O}(N)$
Berndt et. al.	2021	$2^{\mathcal{O}(1/\varepsilon \log(1/\varepsilon)\log(\log(1/\varepsilon)))} + \mathcal{O}(N)$

Main result: faster $(1+\varepsilon)$-approximation for $Q||C_{\max}$



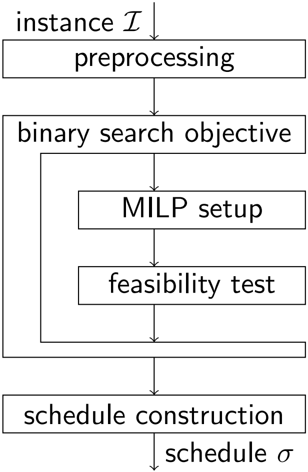
Main result: faster $(1+\varepsilon)$-approximation for $Q||C_{max}$



"Recursive configurations"

- Simulate a new machine as an additional job on a faster machine

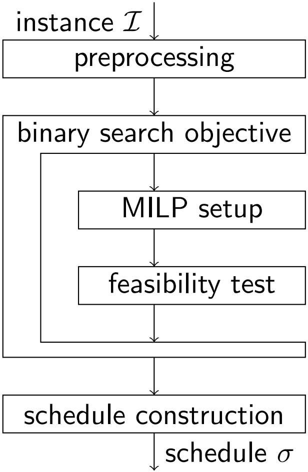
Main result: faster $(1+\varepsilon)$-approximation for $Q||C_{\max}$



"Recursive configurations"

- Simulate a new machine as an additional job on a faster machine
- $\Rightarrow$ MILP formulation with $A_{\max} \in \mathcal{O}(\log(1/\varepsilon))$ (instead of $\mathcal{O}(1/\varepsilon)$).

Main result: faster $(1+\varepsilon)$-approximation for $Q||C_{\max}$



"Recursive configurations"

- Simulate a new machine as an additional job on a faster machine
- $\Rightarrow$ MILP formulation with $A_{\max} \in \mathcal{O}(\log(1/\varepsilon))$ (instead of $\mathcal{O}(1/\varepsilon)$).

Theorem (Linearized support bound)

Any feasible bounded ILP with m constraints and largest column 1-norm $A_{\max}$ has an optimal solution $\boldsymbol{x}$ with $\mathrm{supp}(\boldsymbol{x}) \leq 2m\log(1.46A_{\max})$.

Further results

$Q|HM|C_{\max}$:
(high multiplicity input of jobs *and* machines)

No known approximation scheme

Further results

$Q|HM|C_{\max}$:
(high multiplicity input of jobs *and* machines)

No known approximation scheme $\rightarrow$ Our result (2023):

$2^{\mathcal{O}(1/\varepsilon \log^3(1/\varepsilon)) \log(\log(1/\varepsilon)))} \cdot \langle\mathcal{I}\rangle^{\mathcal{O}(1)}$

Further results

$Q|HM|C_{\max}$:
(high multiplicity input of jobs *and* machines)

No known approximation scheme → Our result (2023):

$2^{\mathcal{O}(1/\varepsilon \log^3(1/\varepsilon)) \log(\log(1/\varepsilon)))} \cdot \langle \mathcal{I} \rangle^{\mathcal{O}(1)}$

$R_K Q||C_{\max}$:
(K types of machines, each with uniform speeds)

Jansen, Maack (2019):
$2^{\mathcal{O}(K \log(K) \cdot 1/\varepsilon^3 \log^5(1/\varepsilon))} + (K \cdot N)^{\mathcal{O}(1)}$

Further results

$Q|HM|C_{\max}$:
(high multiplicity input of jobs *and* machines)

No known approximation scheme → Our result (2023): $2^{\mathcal{O}(1/\varepsilon \log^3(1/\varepsilon)) \log(\log(1/\varepsilon)))} \cdot \langle \mathcal{I} \rangle^{\mathcal{O}(1)}$

$R_K Q||C_{\max}$:
(K types of machines, each with uniform speeds)

Jansen, Maack (2019): $2^{\mathcal{O}(K \log(K) \cdot 1/\varepsilon^3 \log^5(1/\varepsilon))} + (K \cdot N)^{\mathcal{O}(1)}$ → Our result (2023): $2^{\mathcal{O}(K \log(K) \cdot 1/\varepsilon \log^3(1/\varepsilon)) \log(\log(1/\varepsilon)))} + \mathcal{O}(K \cdot N)$

Problem definition

Makespan minimization on uniformly related machines:
Given a set $\mathcal{J}$ of N jobs with processing times $p_1, \ldots, p_n \in \mathbb{N}$, and a set $\mathcal{M}$ of M machines with speeds $s_1, \ldots, s_m \in \mathbb{N}$, find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ which minimizes the makespan

$$C_{\max} := \max_{i \in \mathcal{M}} C_i = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} \frac{p_j}{s_i} \quad .$$

Problem definition

Makespan minimization on uniformly related machines:
Given a set $\mathcal{J}$ of N jobs with processing times $p_1, \ldots, p_n \in \mathbb{N}$, and a set $\mathcal{M}$ of M machines with speeds $s_1, \ldots, s_m \in \mathbb{N}$, find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ which minimizes the makespan

$$C_{\max} := \max_{i \in \mathcal{M}} C_i = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} \frac{p_j}{s_i} \ .$$

Goal: efficient polynomial-time approximation scheme (EPTAS), which, for given $\varepsilon > 0$ and any instance $\mathcal{I}$ in time $f(1/\varepsilon) + \langle \mathcal{I} \rangle^{\mathcal{O}(1)}$ computes a schedule of makespan $C_{\max} \leq (1+\varepsilon)\mathrm{OPT}$.

Problem definition

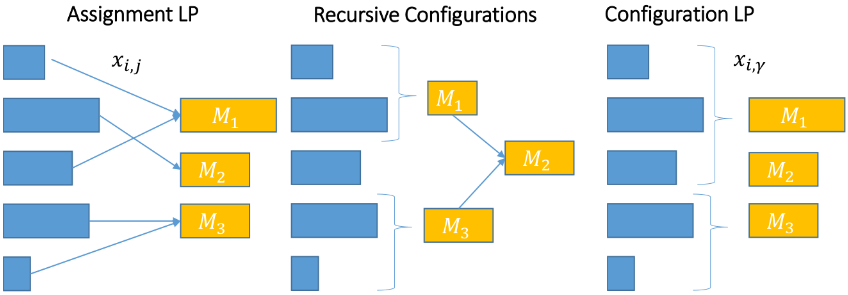
Makespan minimization on uniformly related machines:
Given a set $\mathcal{J}$ of N jobs with processing times $p_1, \ldots, p_n \in \mathbb{N}$, and a set $\mathcal{M}$ of M machines with speeds $s_1, \ldots, s_m \in \mathbb{N}$, find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ which minimizes the makespan

$$C_{\max} := \max_{i \in \mathcal{M}} C_i = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} \frac{p_j}{s_i} .$$

Goal: efficient polynomial-time approximation scheme (EPTAS), which, for given $\varepsilon > 0$ and any instance $\mathcal{I}$ in time $f(1/\varepsilon) + \langle \mathcal{I} \rangle^{\mathcal{O}(1)}$ computes a schedule of makespan $C_{\max} \leq (1+\varepsilon)\mathrm{OPT}$.

$\rightarrow$ EPTAS are fixed-parameter algorithms with parameter $(1/\varepsilon)$
$\rightarrow$ M., van Bevern (2018): "Parameterized complexity of scheduling: 15 open problems"

Problem formulation overview



formulation	assignment	recursive conf.	configurations
jobs/machine	N	$\log(1/\delta)$	$1/\delta$

An EPTAS design technique

Lemma

For any $\delta > 0$ and $\delta \to 0$ it holds that

$$(1 + O(\delta))^{\mathcal{O}(1)} = 1 + \mathcal{O}(\delta) + \mathcal{O}(\delta^2) + \ldots = 1 + \mathcal{O}(\delta) \ .$$

An EPTAS design technique

Lemma

For any $\delta > 0$ and $\delta \to 0$ it holds that

$$(1 + O(\delta))^{\mathcal{O}(1)} = 1 + \mathcal{O}(\delta) + \mathcal{O}(\delta^2) + \ldots = 1 + \mathcal{O}(\delta) \ .$$

$\Rightarrow$ Can add errors of constant multiples of δ constantly many times, and have input independent constant c such that $\delta = \varepsilon / c$.

EPTAS overview

1. Preprocess the instance
    - Remove negligible jobs and machines
    - Binary search for the makespan
    - Round the processing times and machine speeds
2. Solving an MILP formulation
    - Construct an MILP
    - Find a feasible solution
3. Constructing a schedule
    - Round the configuration variables
    - Assign the jobs & configurations

Preprocessing

1) Remove negligibly short jobs and slow machines:

$$p_i \geq p_{\max} \cdot \delta/N \qquad\qquad s_i \geq s_{\max} \cdot \delta/N$$

Preprocessing

1) Remove negligibly short jobs and slow machines:

$$p_i \geq p_{\max} \cdot \delta/N \qquad s_i \geq s_{\max} \cdot \delta/N$$

2) Bound makespan T and perform binary search:

$$p_{\max}/s_{\max} \leq T \leq N \cdot p_{\max}/s_{\max}$$

Preprocessing

1) Remove negligibly short jobs and slow machines:

$$p_i \geq p_{\max} \cdot \delta / N \qquad s_i \geq s_{\max} \cdot \delta / N$$

2) Bound makespan T and perform binary search:

$$p_{\max}/s_{\max} \leq T \leq N \cdot p_{\max}/s_{\max}$$

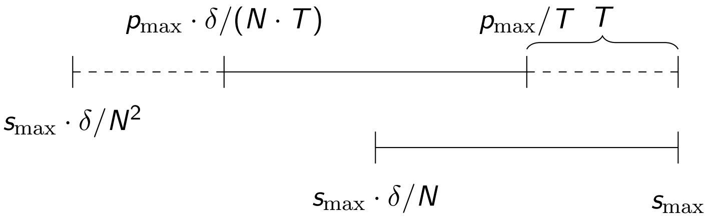
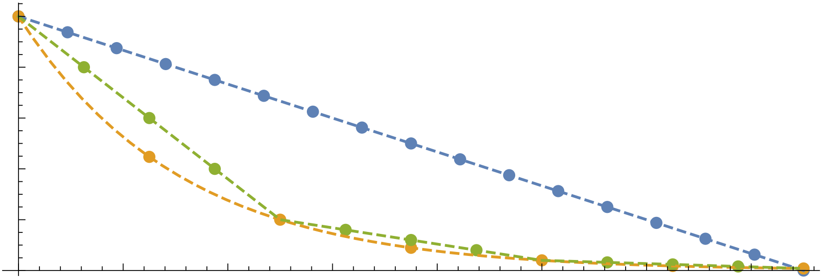


Figure: Overview on the range of parameters.

Rounding scheme



rounding	arithmetic	geometric	geo-arithmetic
points	$1/\delta^2$	$\log_{1+\delta}(1/\delta)$	$1/\delta \log(1/\delta)$
conf. size	2	$1/\delta$	$\log(1/\delta)$
variables	$\mathcal{O}(1/\delta^2)$	$2^{\mathcal{O}(1/\delta \log(1/\delta))}$	$2^{\mathcal{O}(\log^2(1/\delta))}$

Constructing an MILP

$$\sum_{\gamma\in\mathcal{C}_i} x_{i,\gamma} - \mu_i = \sum_{i'=1}^{\tau}\sum_{\gamma\in\mathcal{C}_{i'}} \gamma_i \cdot x_{i',\gamma} - \eta_i \geq 0 \qquad \text{for } i = 1,\dots,\tau$$

$$x_{i,\gamma} \geq 0 \text{ for } i = 1,\dots,\tau, \gamma \in \mathcal{C}_i$$

$$x_{i,\gamma} \in \mathbb{Z}_{\geq 0} \text{ for } i = 1,\dots,L, \gamma \in \mathcal{C}_i \qquad \text{(recursive-MILP)}$$

$x_{i,\gamma}$ number of configurations γ on machines of speed s_i.

$\mathcal{C}_i$: set of configurations for s_i

μ_i : #machines of speed s_i

$\tau \in \mathcal{O}(1/\delta \log(N/\delta))$

γ : a configuration vector

η_i #jobs of processing time s_i

$L \in \mathcal{O}(1/\delta \log(1/\delta))$

Support size bounds for ILPs

Classical result: (Eisenbrand, Shmonin 2004). Any feasible and bounded IP with m constraints admits a solution with support size $s \leq 2mlog(4m\Delta)$, where Δ is the largest absolute value of any entry in the constraint matrix A.

Support size bounds for ILPs

Classical result: (Eisenbrand, Shmonin 2004). Any feasible and bounded IP with m constraints admits a solution with support size $s \leq 2mlog(4m\Delta)$, where Δ is the largest absolute value of any entry in the constraint matrix A.

New main result: Any feasible bounded ILP with an m-row constraint matrix A has an optimal solution with support size $s \leq m \cdot (log(3A_{\max}) + \sqrt{\log(A_{\max})})$, where $A_{\max}$ is the largest 1-norm of any column of A.

Support size bounds for ILPs

Classical result: (Eisenbrand, Shmonin 2004). Any feasible and bounded IP with m constraints admits a solution with support size $s \leq 2mlog(4m\Delta)$, where Δ is the largest absolute value of any entry in the constraint matrix A.

New main result: Any feasible bounded ILP with an m-row constraint matrix A has an optimal solution with support size $s \leq m \cdot (log(3A_{\max}) + \sqrt{\log(A_{\max})})$, where $A_{\max}$ is the largest 1-norm of any column of A.

Our result builds on determinant analysis and Siegel's Lemma (1929) from number theory.

Finding a feasible solution

Lemma (Lenstra, Kannan)

An MILP instance $\mathcal{I}$ with n integral variables, s of which are non-zero, can be solved or proved infeasible with run time:

$$\binom{n}{s} \cdot s^s \cdot \langle \mathcal{I} \rangle^{\mathcal{O}(1)} = 2^{\mathcal{O}(s \log(n))} \cdot \langle \mathcal{I} \rangle^{\mathcal{O}(1)} \quad .$$

Finding a feasible solution

Lemma (Lenstra, Kannan)

An MILP instance $\mathcal{I}$ with n integral variables, s of which are non-zero, can be solved or proved infeasible with run time:

$$\binom{n}{s} \cdot s^s \cdot \langle \mathcal{I} \rangle^{\mathcal{O}(1)} = 2^{\mathcal{O}(s \log(n))} \cdot \langle \mathcal{I} \rangle^{\mathcal{O}(1)} \quad .$$

We have $n \in 2^{\mathcal{O}(\log^2(1/\delta))}$ and $s \in \mathcal{O}(1/\delta \log(1/\delta) \log(\log(1/\delta)))$.

$$\Rightarrow \quad 2^{\mathcal{O}(1/\delta \log^3(1/\delta) \log(\log(1/\delta)))} \cdot \log^{\mathcal{O}(1)}(N)$$

Constructing a schedule

- Make all $x_{i,\gamma}$ integral.
  - Use vertex solution of fractional part of recursive-MILP.
  - For a machine speed $\mathcal{O}(1/\delta \log(1/\delta))$ many pos. variables.
  - Round down, loss geometric sum, small on fastest machine.
- Recursively construct a schedule, resolving virtual machines.

Faster Schedule Construction

So far: EPTAS for $Q||C_{\max}$ with almost linear run time $2^{O(1/\varepsilon \log^3(1/\varepsilon) \log(\log(1/\varepsilon)))} + O(N \log^2(N))$.

Faster Schedule Construction

So far: EPTAS for $Q||C_{\max}$ with almost linear run time $2^{O(1/\varepsilon \log^3(1/\varepsilon) \log(\log(1/\varepsilon)))} + O(N \log^2(N))$.

Bottleneck with respect to N: transforming MILP solution to feasible schedule.

Faster Schedule Construction

So far: EPTAS for $Q||C_{\max}$ with almost linear run time $2^{O(1/\varepsilon \log^3(1/\varepsilon) \log(\log(1/\varepsilon)))} + O(N \log^2(N))$.
Bottleneck with respect to N: transforming MILP solution to feasible schedule.

- conventional MILP formulation (hybrid-MILP) using both configuration and assignment variables to improve the run time in N.
- First, transform solution of (recursive-MILP) into a solution of (hybrid-MILP) in sublinear time in N
- Then, we construct schedule from solution to (hybrid-MILP) in linear time.

Faster Schedule Construction

So far: EPTAS for $Q||C_{\max}$ with almost linear run time $2^{O(1/\varepsilon \log^3(1/\varepsilon) \log(\log(1/\varepsilon)))} + O(N \log^2(N))$.

Bottleneck with respect to N: transforming MILP solution to feasible schedule.

- conventional MILP formulation (hybrid-MILP) using both configuration and assignment variables to improve the run time in N.
- First, transform solution of (recursive-MILP) into a solution of (hybrid-MILP) in sublinear time in N
- Then, we construct schedule from solution to (hybrid-MILP) in linear time.

This transforms a solution of (recursive-MILP) into a valid schedule in time linear in N.

Hybrid-MILP

Then (hybrid-MILP) is has no recursive configurations.

Hybrid-MILP

Then (hybrid-MILP) is has no recursive configurations.

Variables:

- configuration variables $x_{i,\gamma}$ that indicate how often a configuration γ is used on machine i

Hybrid-MILP

Then (hybrid-MILP) is has no recursive configurations.

Variables:

- configuration variables $x_{i,\gamma}$ that indicate how often a configuration γ is used on machine i
- handle short jobs (with processing time $\leq \delta/s_i$ on machine i) via assignment variables $y_{i,j}$ indicating how many jobs of processing time p_j are assigned to machines of speed s_i.

Hybrid-MILP

Then (hybrid-MILP) is has no recursive configurations.

Variables:

- configuration variables $x_{i,\gamma}$ that indicate how often a configuration γ is used on machine i
- handle short jobs (with processing time $\leq \delta/s_i$ on machine i) via assignment variables $y_{i,j}$ indicating how many jobs of processing time p_j are assigned to machines of speed s_i.

Constraints:

- First set of constraints enforce that every machine is assigned a configuration.

Hybrid-MILP

Then (hybrid-MILP) is has no recursive configurations.

Variables:

- configuration variables $x_{i,\gamma}$ that indicate how often a configuration γ is used on machine i
- handle short jobs (with processing time $\leq \delta/s_i$ on machine i) via assignment variables $y_{i,j}$ indicating how many jobs of processing time p_j are assigned to machines of speed s_i.

Constraints:

- First set of constraints enforce that every machine is assigned a configuration.
- Second set of constraints guarantee that every job is scheduled somewhere.

Hybrid-MILP

Then (hybrid-MILP) is has no recursive configurations.

Variables:

- configuration variables $x_{i,\gamma}$ that indicate how often a configuration γ is used on machine i
- handle short jobs (with processing time $\leq \delta/s_i$ on machine i) via assignment variables $y_{i,j}$ indicating how many jobs of processing time p_j are assigned to machines of speed s_i.

Constraints:

- First set of constraints enforce that every machine is assigned a configuration.
- Second set of constraints guarantee that every job is scheduled somewhere.
- Third set constraints ensure that the speed used by short jobs is at most the speed left free by configurations.

Constructing a schedule from Hybrid-MILP

Step 1: Convert an optimal solution $x^\star$ of (recursive-MILP) into a feasible solution (x, y) of (hybrid-MILP) in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} \log^{\mathcal{O}(1)}(N)$.

Constructing a schedule from Hybrid-MILP

Step 1: Convert an optimal solution $x^\star$ of (recursive-MILP) into a feasible solution (x, y) of (hybrid-MILP) in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} \log^{\mathcal{O}(1)}(N)$.

Step 2: From feasible solution of (hybrid-MILP), construct schedule with makespan at most $(1 + \mathcal{O}(\delta))T$ in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} + \mathcal{O}(N)$.

Constructing a schedule from Hybrid-MILP

Step 1: Convert an optimal solution $x^\star$ of (recursive-MILP) into a feasible solution (x, y) of (hybrid-MILP) in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} \log^{\mathcal{O}(1)}(N)$.

Step 2: From feasible solution of (hybrid-MILP), construct schedule with makespan at most $(1 + \mathcal{O}(\delta))T$ in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} + \mathcal{O}(N)$.

Key ideas:

- round configuration variables down and assign one configuration to a fastest machine for every rounded variable

Constructing a schedule from Hybrid-MILP

Step 1: Convert an optimal solution $x^\star$ of (recursive-MILP) into a feasible solution (x, y) of (hybrid-MILP) in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} \log^{\mathcal{O}(1)}(N)$.

Step 2: From feasible solution of (hybrid-MILP), construct schedule with makespan at most $(1 + \mathcal{O}(\delta))T$ in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} + \mathcal{O}(N)$.

Key ideas:

- round configuration variables down and assign one configuration to a fastest machine for every rounded variable
- by use of basic solutions, we construct a schedule with small multiplicative error at most $(1 + \mathcal{O}(\delta))T$

Constructing a schedule from Hybrid-MILP

Step 1: Convert an optimal solution $x^\star$ of (recursive-MILP) into a feasible solution (x, y) of (hybrid-MILP) in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} \log^{\mathcal{O}(1)}(N)$.

Step 2: From feasible solution of (hybrid-MILP), construct schedule with makespan at most $(1 + \mathcal{O}(\delta))T$ in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} + \mathcal{O}(N)$.

Key ideas:

- round configuration variables down and assign one configuration to a fastest machine for every rounded variable
- by use of basic solutions, we construct a schedule with small multiplicative error at most $(1 + \mathcal{O}(\delta))T$
- assign any machine speed at most 2 fractional assignment variables from short jobs, as variables only become fractional when preceding or current group runs out of machine capacity

Constructing a schedule from Hybrid-MILP

Step 1: Convert an optimal solution $x^\star$ of (recursive-MILP) into a feasible solution (x, y) of (hybrid-MILP) in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} \log^{\mathcal{O}(1)}(N)$.

Step 2: From feasible solution of (hybrid-MILP), construct schedule with makespan at most $(1+\mathcal{O}(\delta))T$ in time $2^{\mathcal{O}(1/\delta \log^2(1/\delta))} + \mathcal{O}(N)$.

Key ideas:

- round configuration variables down and assign one configuration to a fastest machine for every rounded variable
- by use of basic solutions, we construct a schedule with small multiplicative error at most $(1+\mathcal{O}(\delta))T$
- assign any machine speed at most 2 fractional assignment variables from short jobs, as variables only become fractional when preceding or current group runs out of machine capacity
- finally, pack all remaining (short) jobs greedily

Outlook: $R_K Q||C_{\max}$

Definition ($R_K Q||C_{\max}$ - Jansen, Maack)

Given M machines $\mathcal{M}$ with speeds s_i, and type k and N jobs $\mathcal{J}$ with processing times $p_{i,j}$, find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ minimizing:

$$C_{\max} := \max_{i \in \mathcal{M}} C_i = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(j)} \frac{p_{i,j}}{s_i}$$

Outlook: $R_K Q||C_{\max}$

Definition ($R_K Q||C_{\max}$ - Jansen, Maack)

Given M machines $\mathcal{M}$ with speeds s_i, and type k and N jobs $\mathcal{J}$ with processing times $p_{i,j}$, find a schedule $\sigma : \mathcal{J} \to \mathcal{M}$ minimizing:

$$C_{\max} := \max_{i \in \mathcal{M}} C_i = \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(j)} \frac{p_{i,j}}{s_i}$$

Theorem

There is an EPTAS for $R_K Q||C_{\max}$ with run time

$$2^{\mathcal{O}(K \log(K) 1/\delta \log^3(1/\delta) \log(\log(1/\delta)))} + \mathcal{O}(K \cdot N) \ .$$

Summary

- Integer Linear Programming
  - Any feasible bounded ILP with an m-row constraint matrix A has an optimal solution with support size $s \leq m \cdot (log(3A_{\max}) + \sqrt{\log(A_{\max})})$, where $A_{\max}$ is the largest 1-norm of any column of A.
- $Q||C_{\max}$ results
  - support bound run time improvement
  - simple recursive configurations formulation
  - Rohwedder general MILP algorithms

Summary

- Integer Linear Programming
  - Any feasible bounded ILP with an m-row constraint matrix A has an optimal solution with support size $s \leq m \cdot (log(3A_{max}) + \sqrt{\log(A_{max})})$, where A_{max} is the largest 1-norm of any column of A.
- $Q||C_{max}$ results
  - support bound run time improvement
  - simple recursive configurations formulation
  - Rohwedder general MILP algorithms
- $R_K Q||C_{max}$ results
  - significant run time improvement
  - direct generalization of $Q||C_{max}$

Summary

- Integer Linear Programming
  - Any feasible bounded ILP with an *m*-row constraint matrix *A* has an optimal solution with support size $s \leq m \cdot (log(3A_{\max}) + \sqrt{\log(A_{\max})})$, where $A_{\max}$ is the largest 1-norm of any column of *A*.
- $Q||C_{\max}$ results
  - support bound run time improvement
  - simple recursive configurations formulation
  - Rohwedder general MILP algorithms
- $R_K Q||C_{\max}$ results
  - significant run time improvement
  - direct generalization of $Q||C_{\max}$

https://arxiv.org/abs/2305.08432