Complexity of Scheduling Few Types of Jobs on Related and Unrelated Machines

Martin Koutecký and **Johannes Zink**

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG
Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität
Institut für Informatik

Complexity of Scheduling Few Types of Jobs on Related and Unrelated Machines

Martin Koutecký and **Johannes Zink**

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Introduction: Scheduling

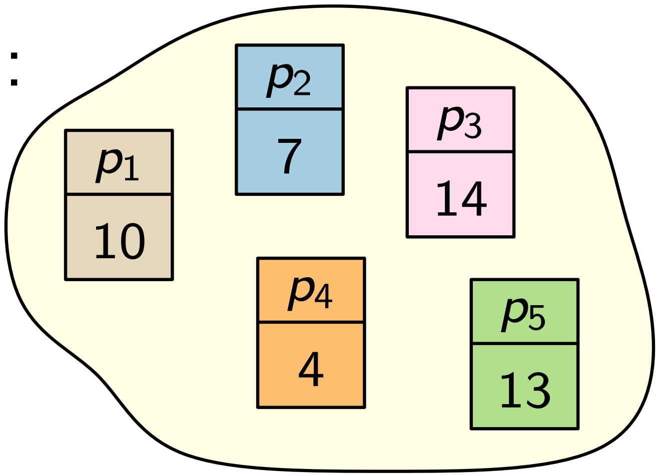
SCHEDULING old fundamental task in combinatorial optimization

Given:

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:

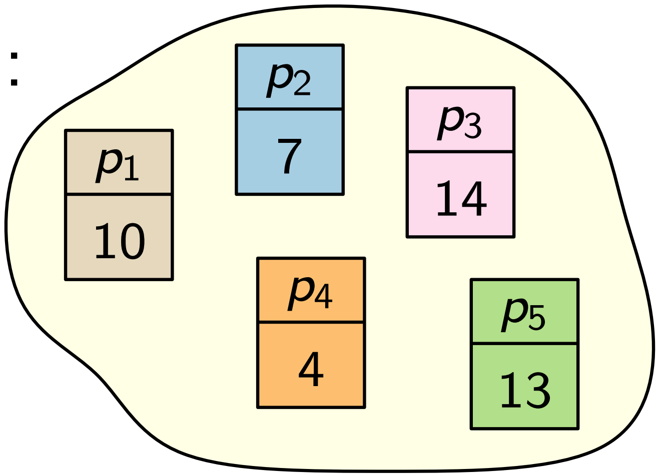


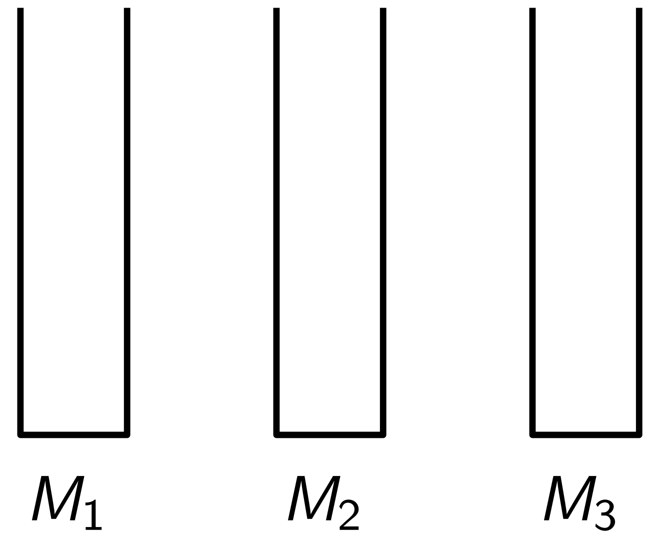
set of n jobs with individual processing times

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





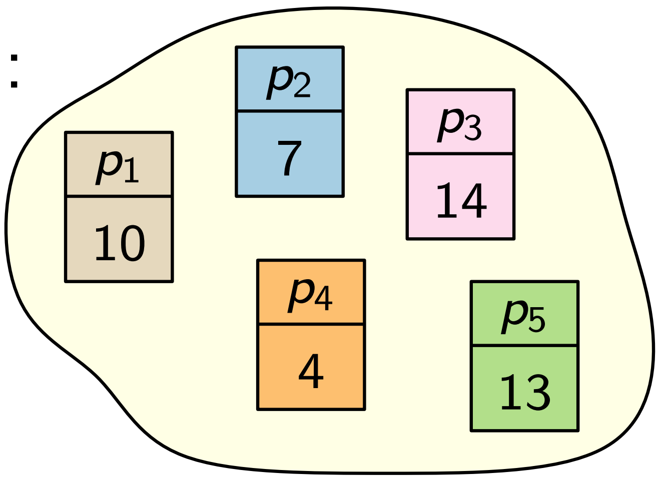
set of n jobs with individual processing times

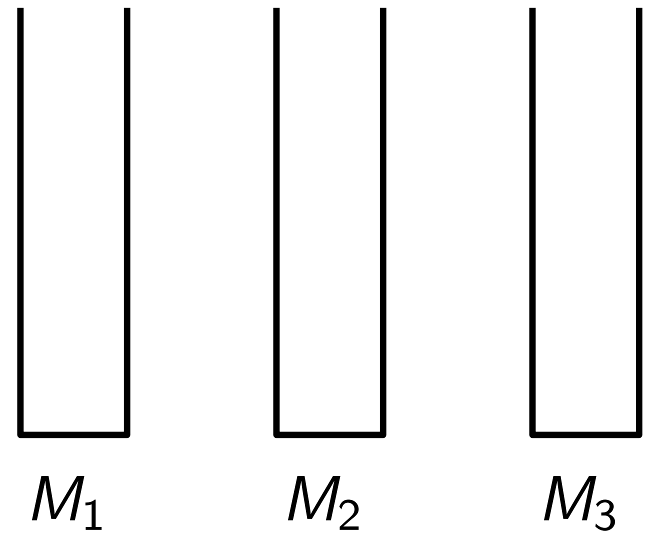
m identical machines

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





set of n jobs with individual processing times

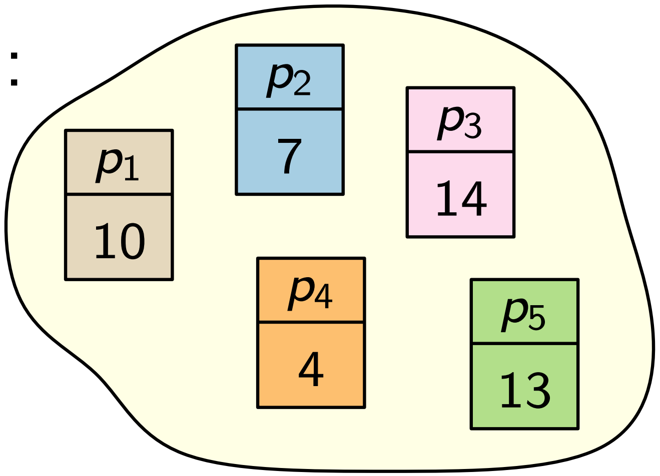
m identical machines

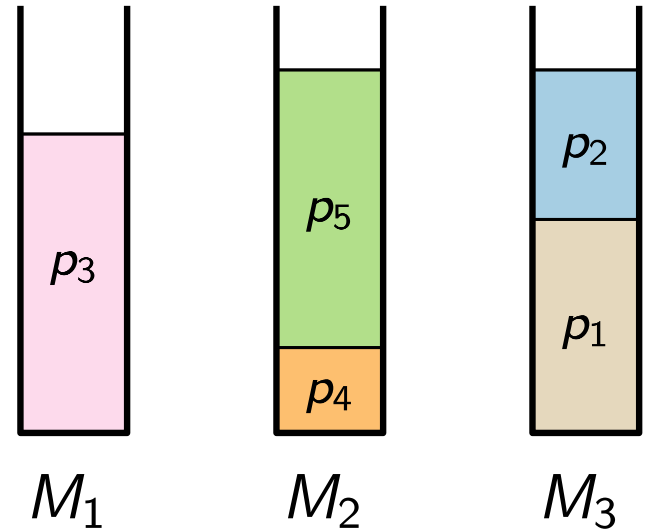
Obj.: find assignment jobs $\rightarrow$ machines, min. completion time

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





set of n jobs with individual processing times

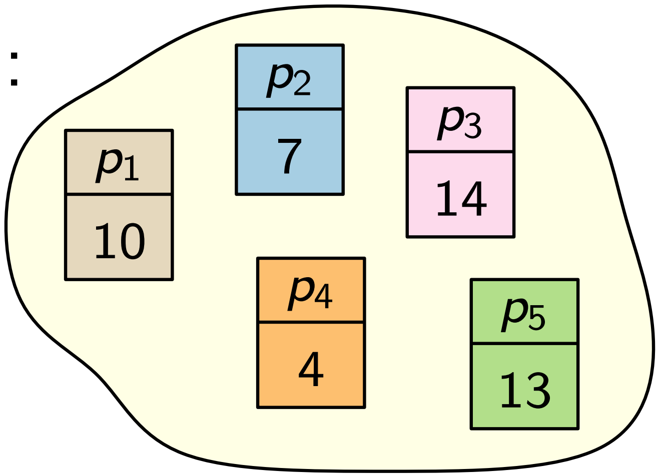
m identical machines

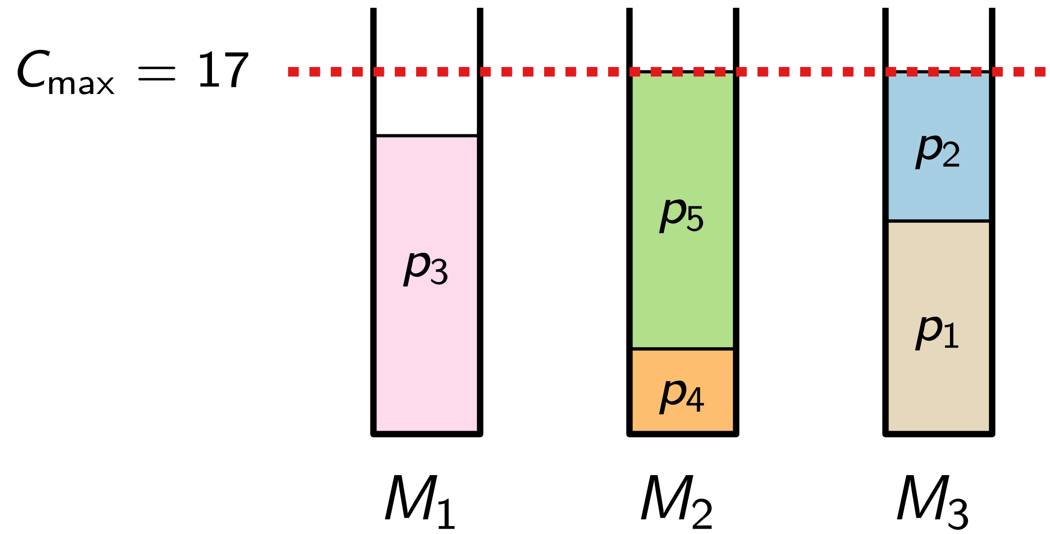
Obj.: find assignment jobs $\rightarrow$ machines, min. completion time

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





set of n jobs with individual processing times

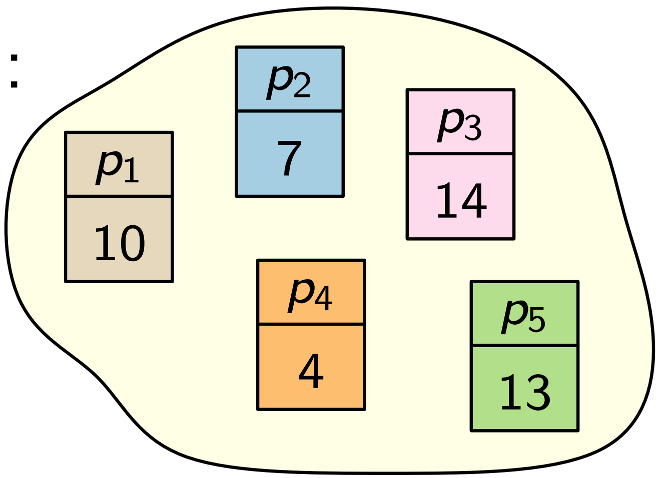
m identical machines

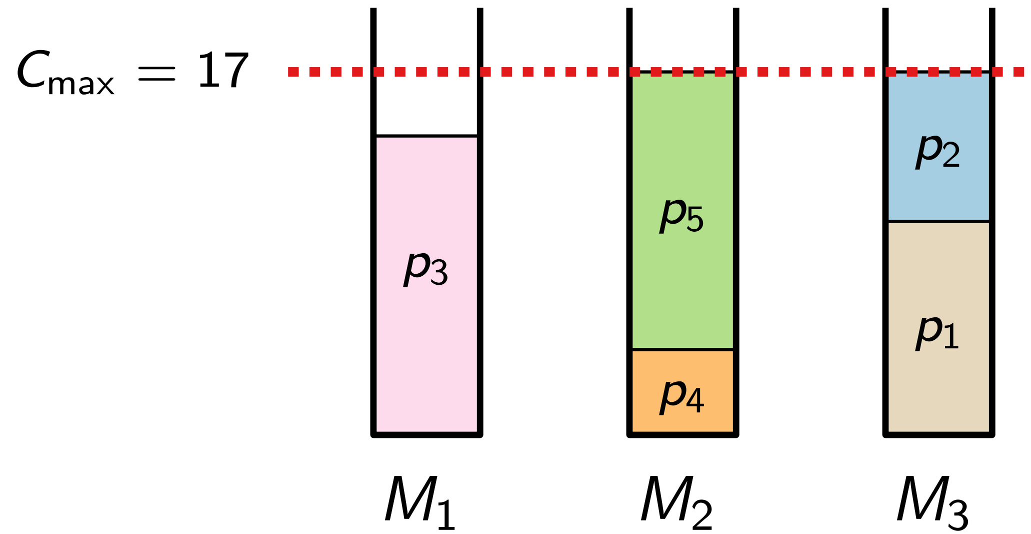
Obj.: find assignment jobs $\rightarrow$ machines, min. completion time

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





set of n jobs with individual processing times

m identical machines

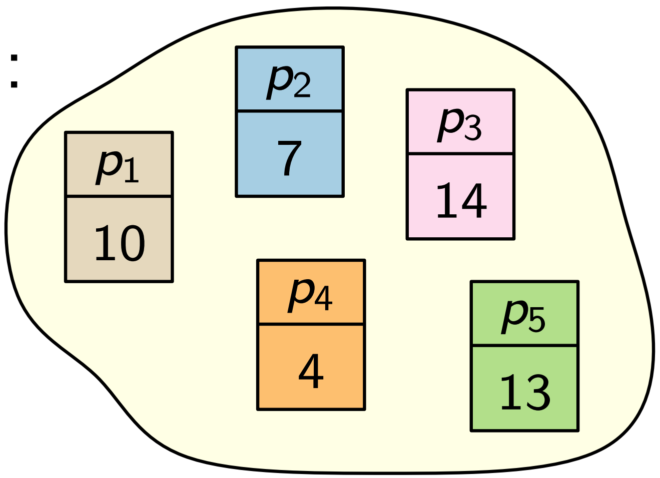
Obj.: find assignment jobs $\rightarrow$ machines, min. completion time

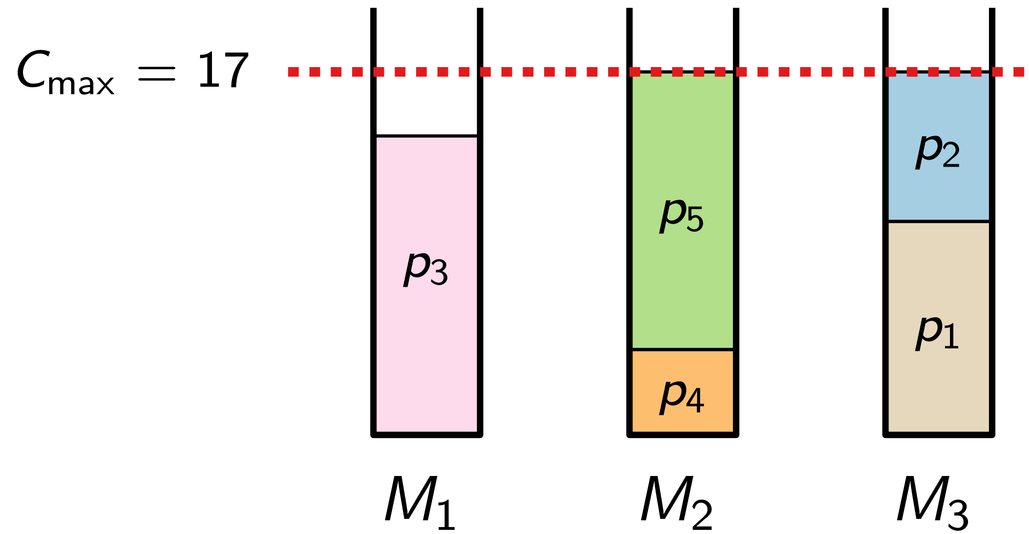
$P||C_{\max}$

Introduction: Scheduling

Scheduling old fundamental task in combinatorial optimization

Given:





set of n jobs with individual processing times

m identical machines

Obj.: find assignment jobs $\to$ machines, min. completion time

$P||C_{\max}$

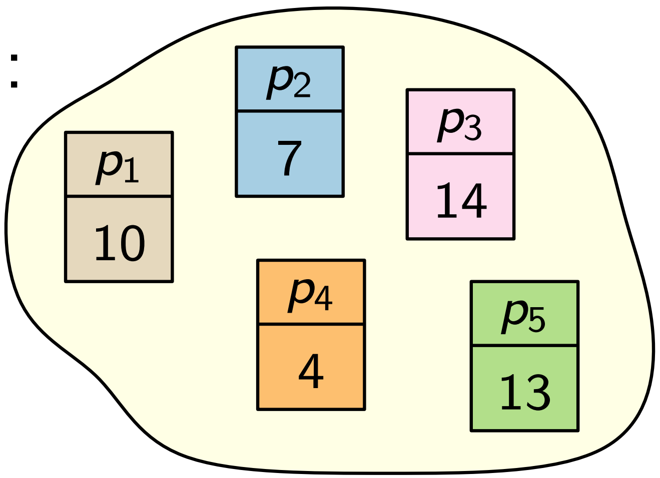
makespan minimization

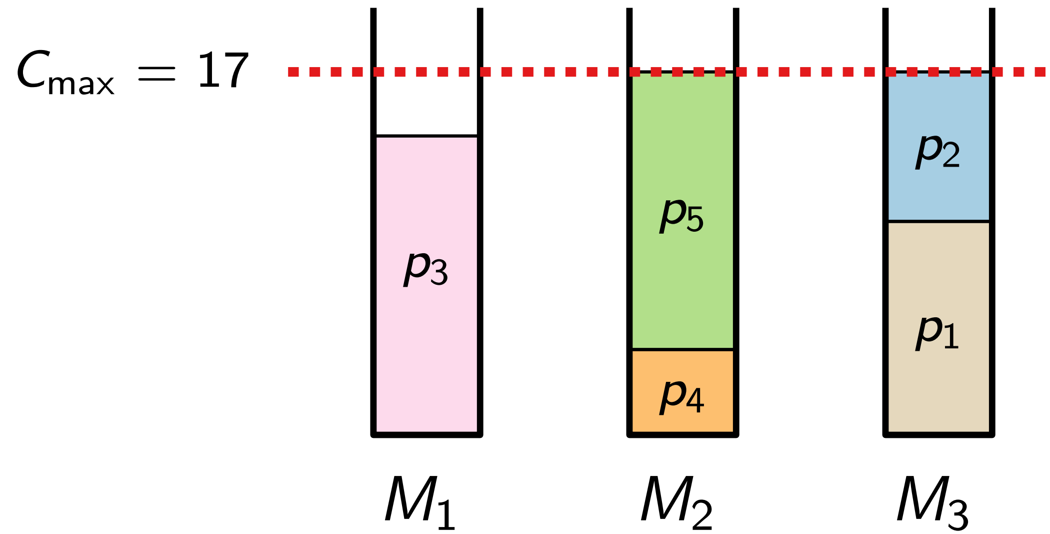
identical machines

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





set of n jobs with individual processing times

m identical machines

Obj.: find assignment jobs $\to$ machines, min. completion time

$P||C_{\max}$ is NP-complete by reduction from BIN PACKING.

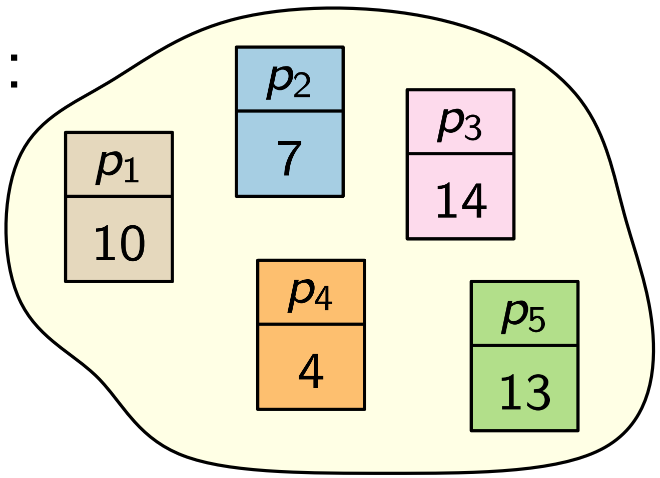
makespan minimization

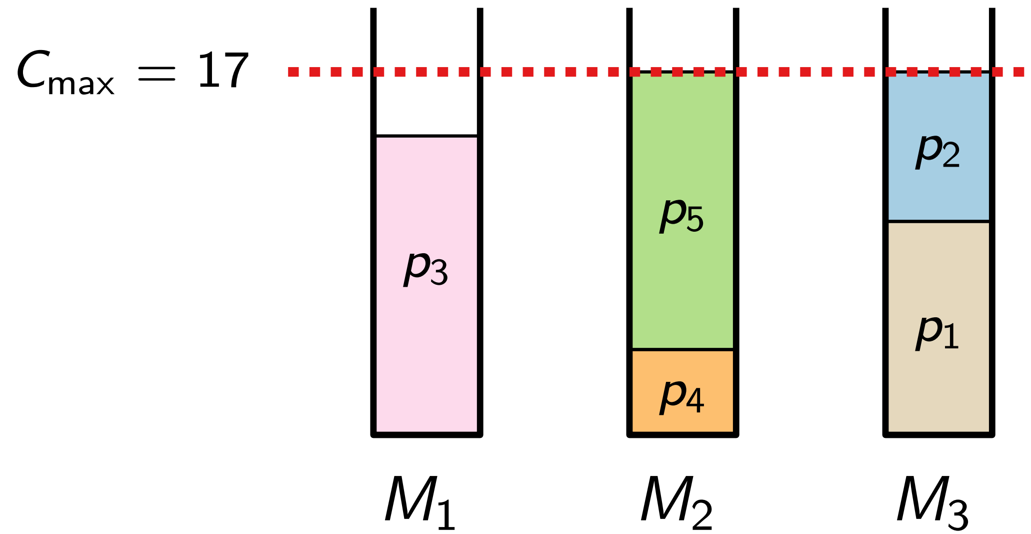
identical machines

Introduction: Scheduling

Scheduling old fundamental task in combinatorial optimization

Given:





natural numbers

set of n ~~jobs with individual processing times~~

m identical machines

Obj.: find assignment jobs $\rightarrow$ machines, min. completion time

$P||C_{\max}$ is NP-complete by reduction from Bin Packing.

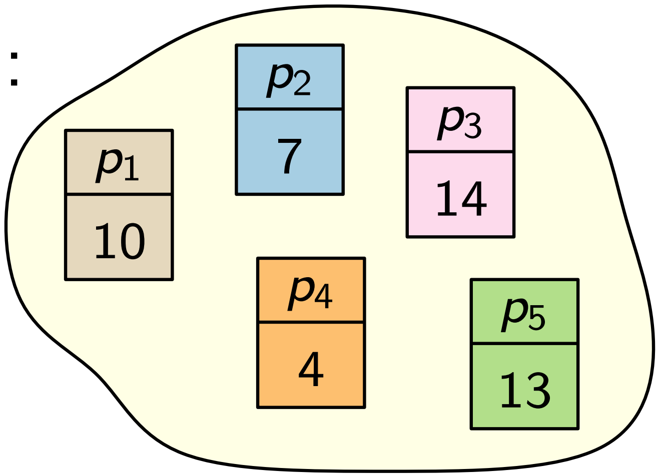
makespan minimization

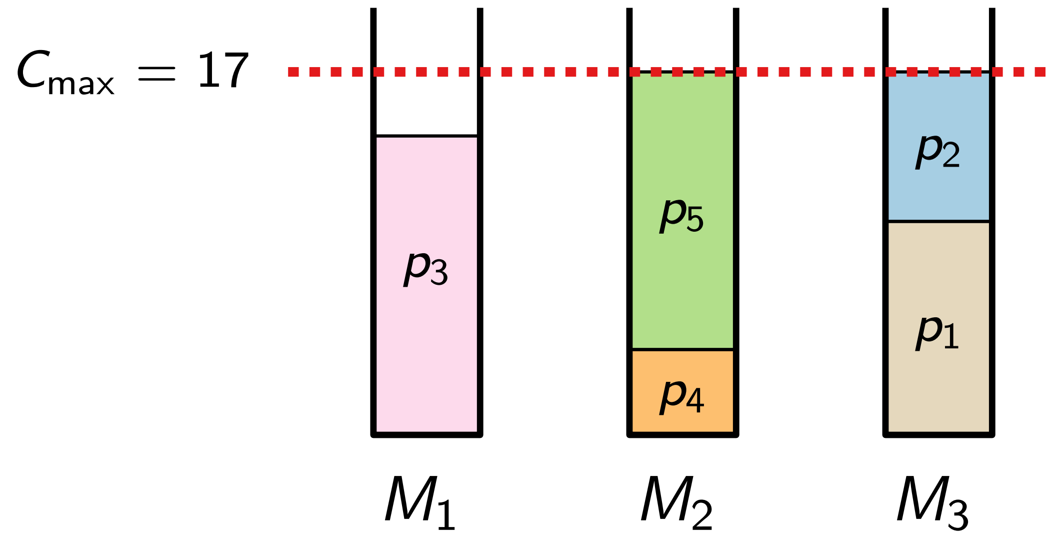
identical machines

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





natural numbers
set of n ~~jobs with individual processing times~~

bins of size $C_{\max}$
m ~~identical machines~~

Obj.: find assignment jobs $\rightarrow$ machines, min. completion time

$P||C_{\max}$ is NP-complete by reduction from BIN PACKING.

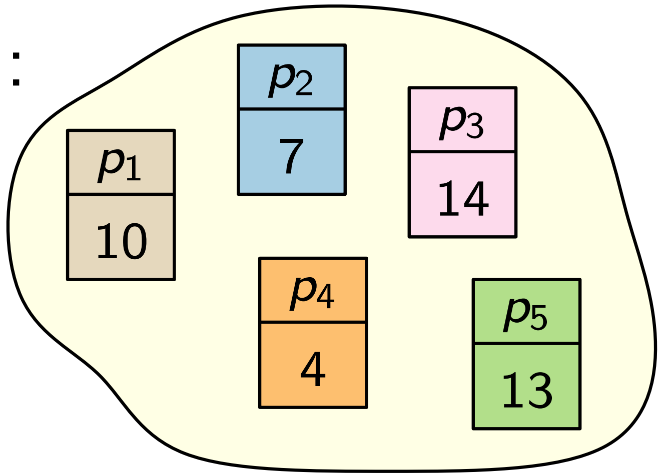
makespan minimization

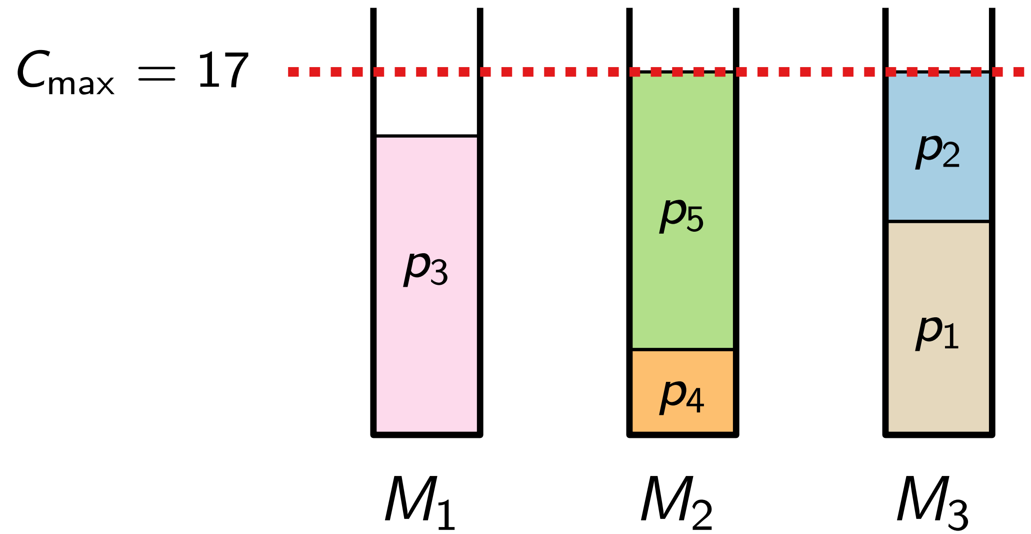
identical machines

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





natural numbers

set of n ~~jobs with individual processing times~~

bins of size $C_{\max}$

m ~~identical machines~~

numbers $\rightarrow$ bins

Obj.: find assignment ~~jobs $\rightarrow$ machines~~, min. completion time

$\boxed{P||C_{\max} \text{ is NP-complete by reduction from } \textsc{Bin Packing}.}$

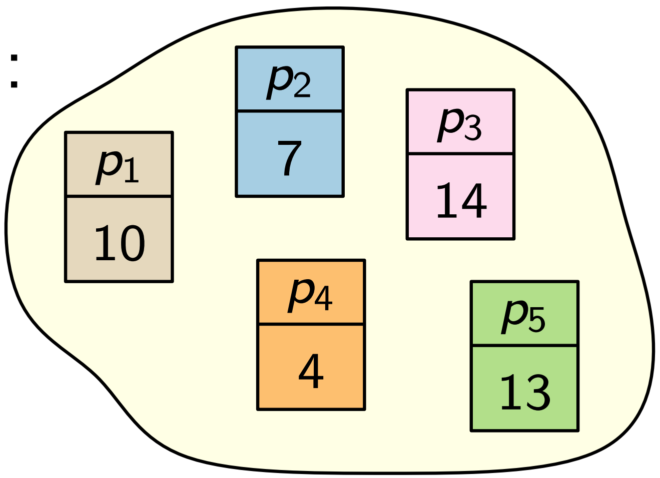
makespan minimization

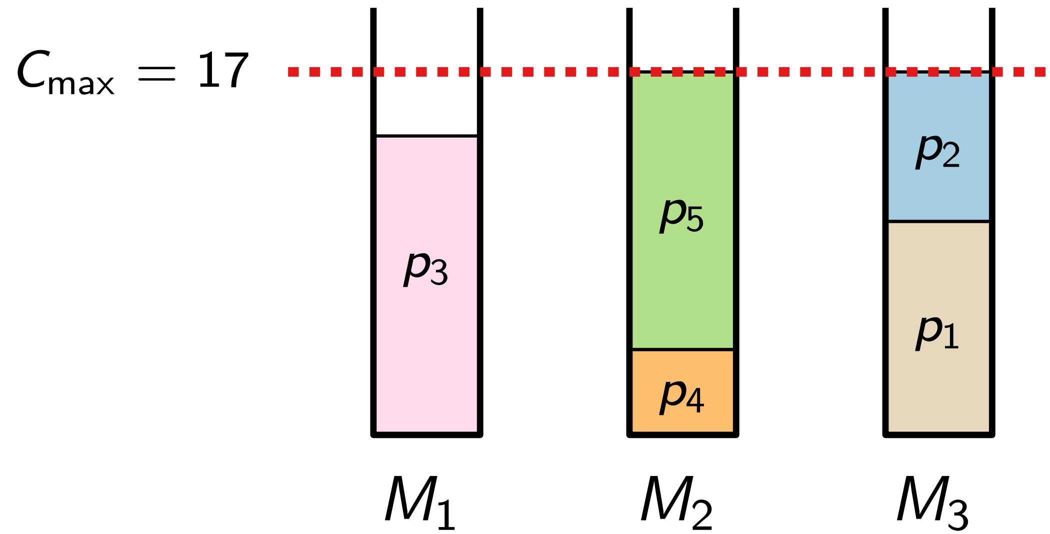
identical machines

Introduction: Scheduling

SCHEDULING old fundamental task in combinatorial optimization

Given:





natural numbers — bins of size $C_{\max}$

set of n ~~jobs with individual processing times~~ — m ~~identical machines~~

numbers $\rightarrow$ bins — no bin overfull

Obj.: find assignment ~~jobs $\rightarrow$ machines~~, ~~min. completion time~~

$\boxed{P||C_{\max} \text{ is NP-complete by reduction from BIN PACKING.}}$

makespan minimization

identical machines

Introduction: Scheduling Few Job Types

So how to simplify our model? What about . . .

Introduction: Scheduling Few Job Types

So how to simplify our model? What about . . .

. . . having few machines?

Introduction: Scheduling Few Job Types

So how to simplify our model? What about . . .

. . . having few machines? No, does not help!

BIN PACKING NP-hard f. const. #bins, even for 2 (PARTITION)

Introduction: Scheduling Few Job Types

So how to simplify our model? What about . . .

. . . having few machines? No, does not help!
BIN PACKING NP-hard f. const. #bins, even for 2 (PARTITION)

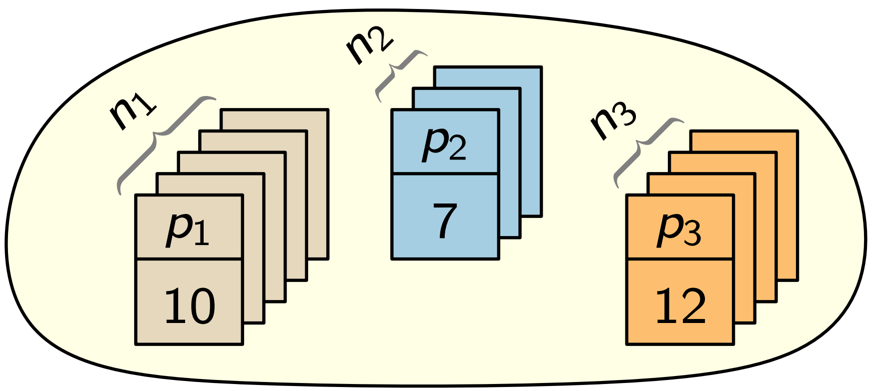
. . . having many duplicates of only few job types?

Introduction: Scheduling Few Job Types

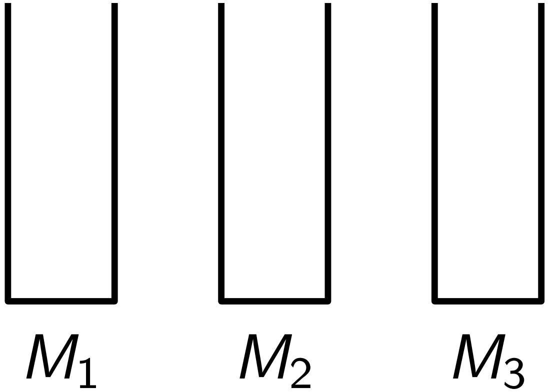
So how to simplify our model? What about . . .

. . . having few machines? No, does not help!
Bin Packing NP-hard f. const. #bins, even for 2 (Partition)

. . . having many duplicates of only few job types?



k job types with $n_1, \ldots, n_k$ identical jobs each



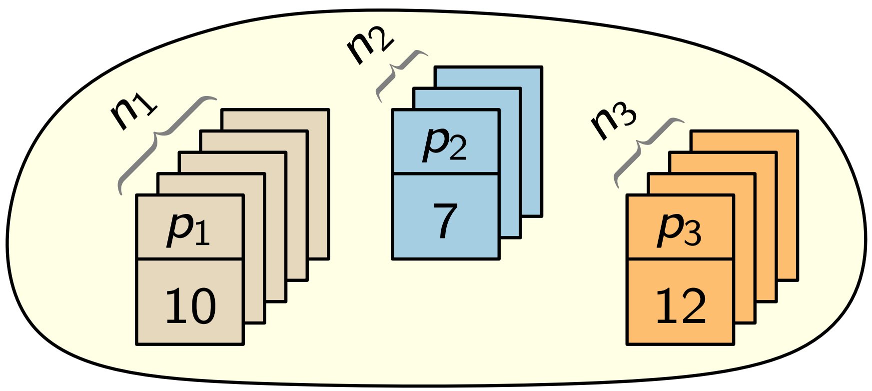
m identical machines

Introduction: Scheduling Few Job Types

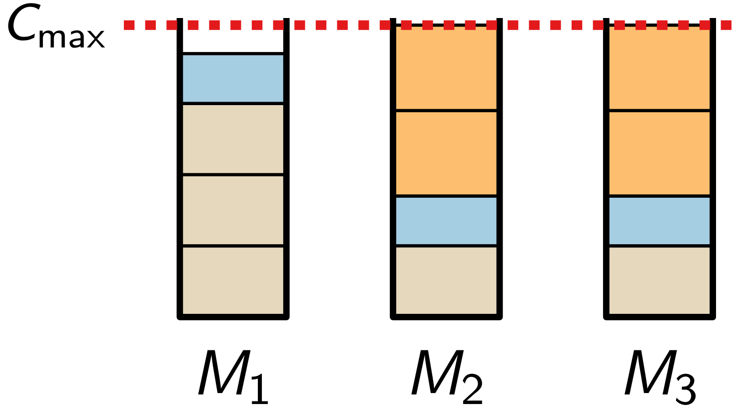
So how to simplify our model? What about ...

... having few machines? No, does not help!
BIN PACKING NP-hard f. const. #bins, even for 2 (PARTITION)

... having many duplicates of only few job types?



k job types with $n_1, \ldots, n_k$ identical jobs each



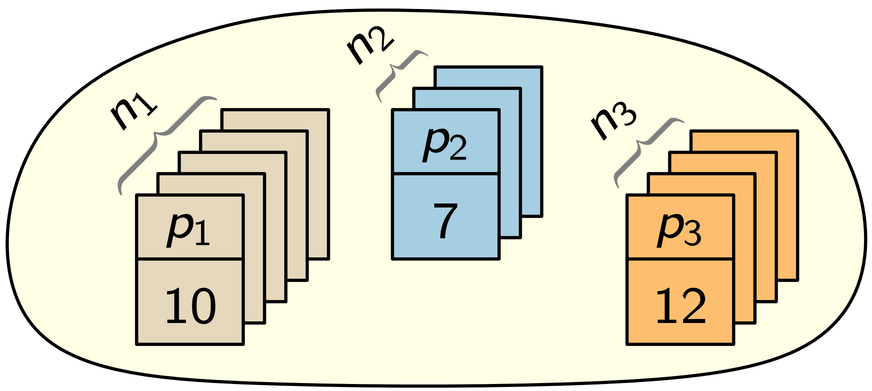
m identical machines

Introduction: Scheduling Few Job Types

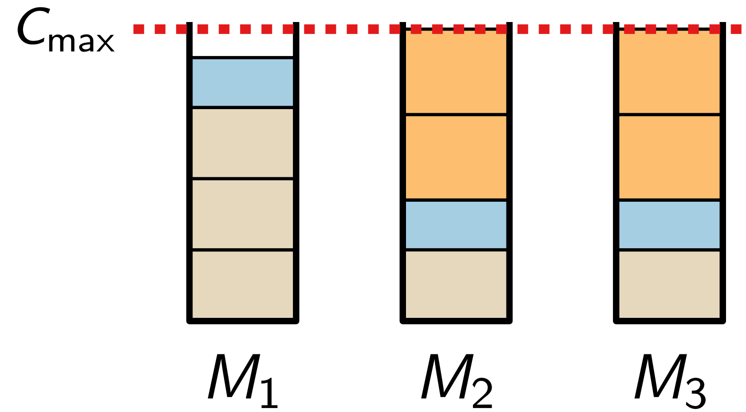
So how to simplify our model? What about ...

... having few machines? No, does not help!
Bin Packing NP-hard f. const. #bins, even for 2 (Partition)

... having many duplicates of only few job types?



k job types with $n_1, \ldots, n_k$ identical jobs each



m identical machines

$P|HM|C_{\max}$

identical machines

high multiplicity of jobs

makespan minimization

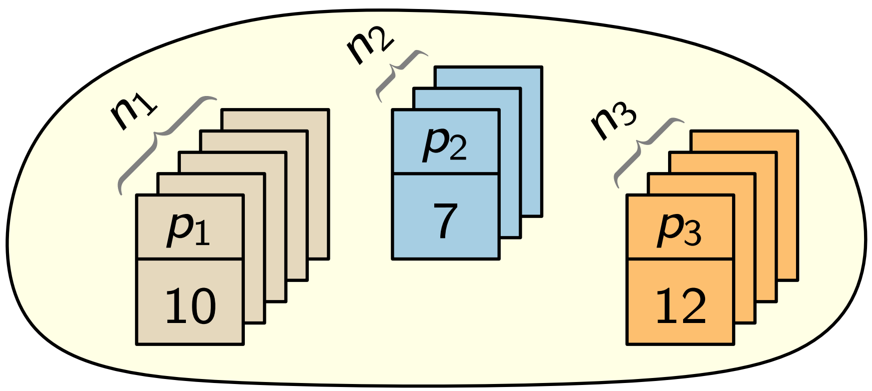
Introduction: Scheduling Few Job Types

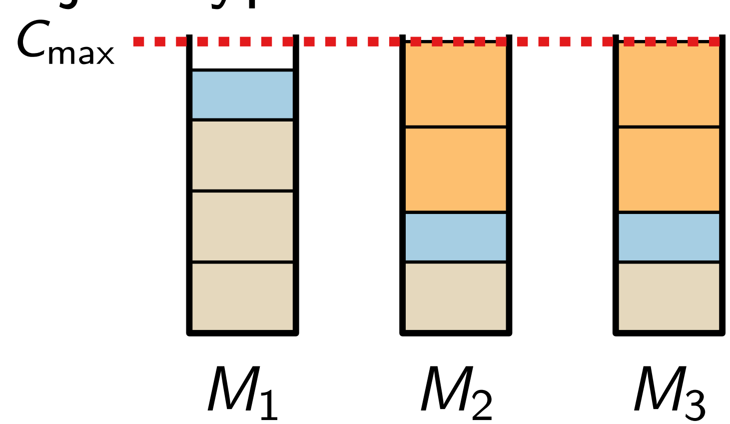
So how to simplify our model? What about ...

... having few machines? No, does not help!

Bin Packing NP-hard f. const. #bins, even for 2 (Partition)

... having many duplicates of only few job types?





k job types with $n_1, \ldots, n_k$ identical jobs each

m identical machines

$P|HM|C_{\max}$ is poly. time solvable (k const.) [Goemans, Rothvoss '14].

makespan minimization

identical machines

high multiplicity of jobs

Scheduling Models

$* | * | C_{\mathsf{max}}$: objective function is makespan minimization

Scheduling Models

$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

Scheduling Models

$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

Scheduling Models

$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

$Q|*|*$: uniformly related machines

Scheduling Models

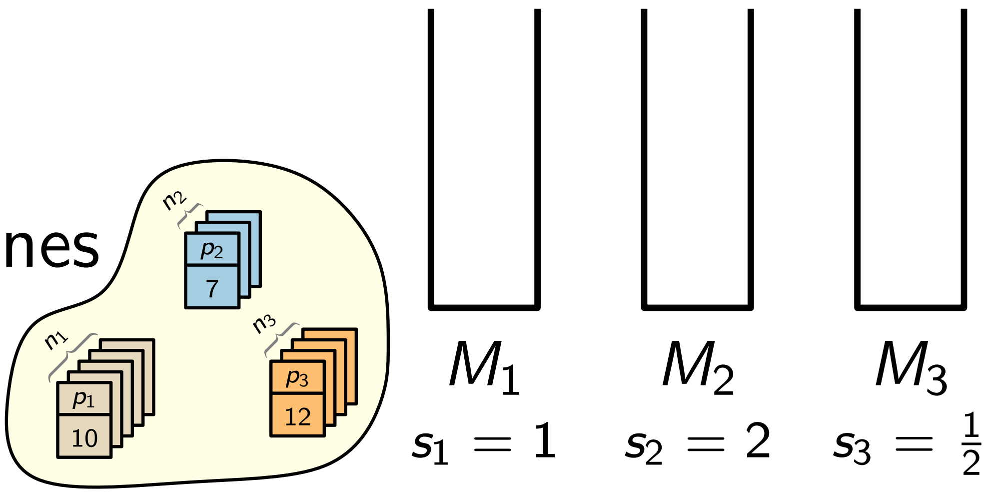
$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

$Q|*|*$: uniformly related machines

- each machine M_i has a speed s_i
- needs time p_j/s_i



Scheduling Models

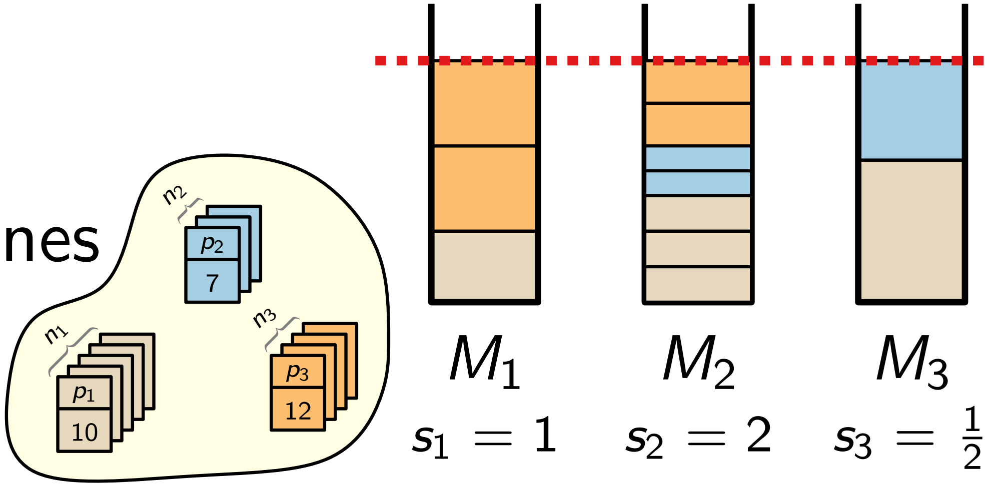
$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

$Q|*|*$: uniformly related machines

- each machine M_i has a speed s_i
- needs time p_j/s_i



Scheduling Models

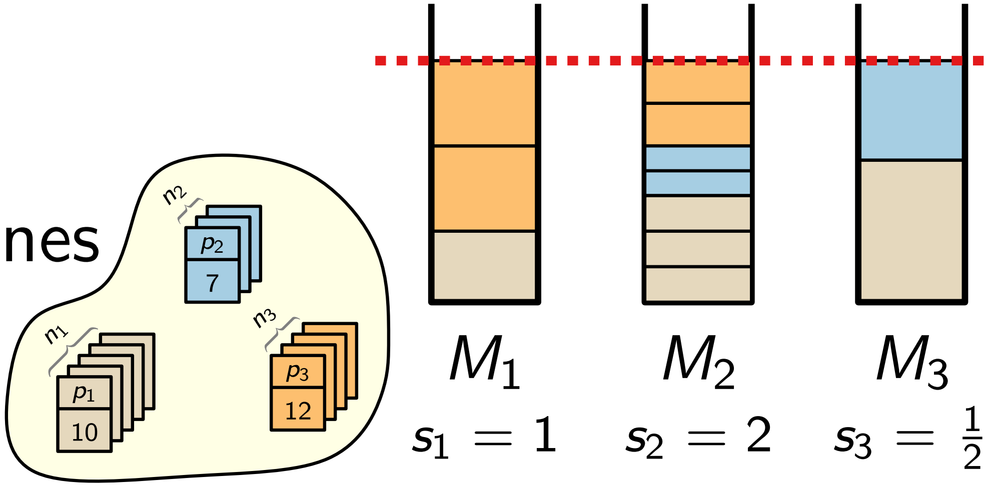
$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

$Q|*|*$: uniformly related machines

- each machine M_i has a speed s_i
- needs time p_j/s_i



$R|*|*$: unrelated machines

Scheduling Models

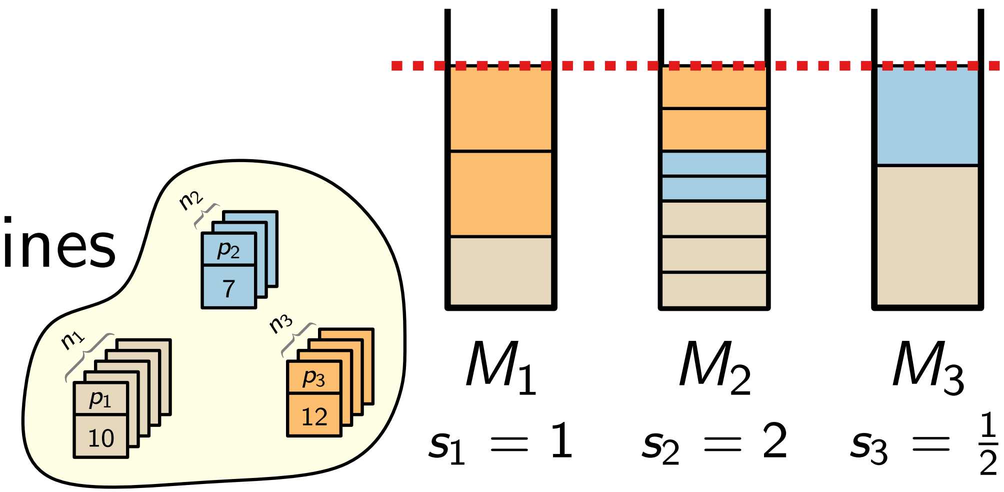
$*|*|C_{\max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

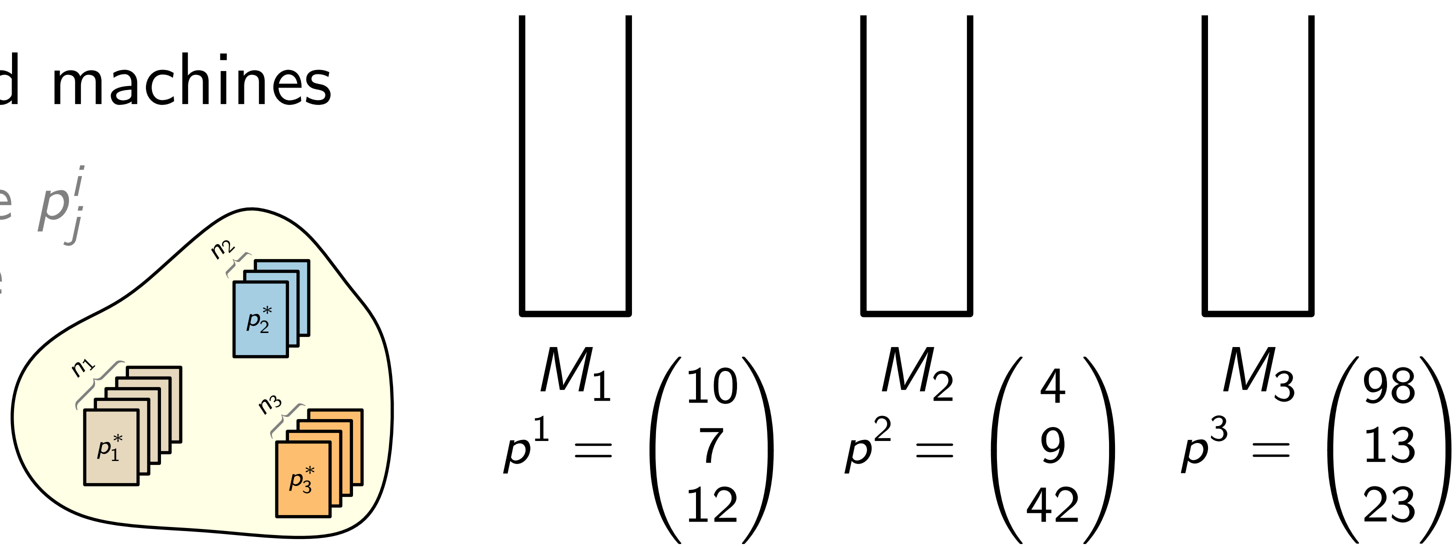
$Q|*|*$: uniformly related machines

- each machine M_i has a speed s_i
- needs time p_j/s_i



$R|*|*$: unrelated machines

- a processing time p_j^i for each machine and each job



Scheduling Models

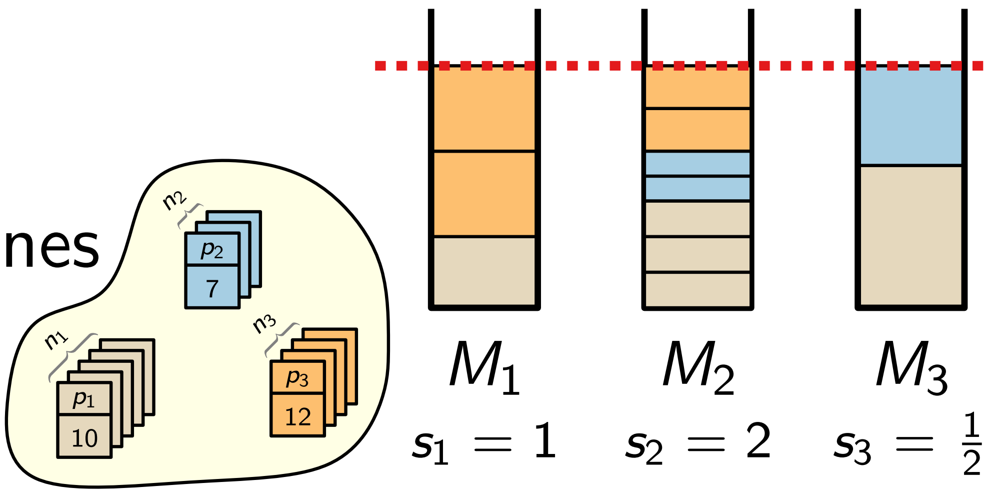
$*|*|C_{max}$: objective function is makespan minimization

$*|HM|*$: high multiplicity of identical jobs

$P|*|*$: identical machines

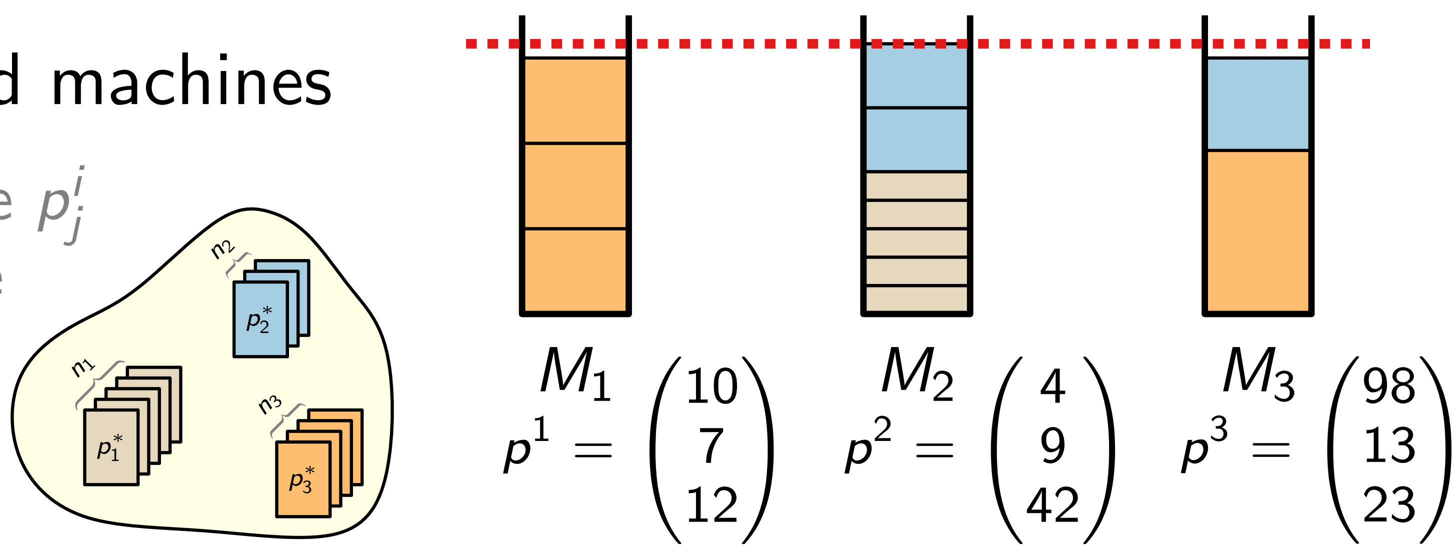
$Q|*|*$: uniformly related machines

- each machine M_i has a speed s_i
- needs time p_j / s_i



$R|*|*$: unrelated machines

- a processing time p_j^i for each machine and each job



Parametrized Complexity

Parametrized Complexity

- A problem is *fixed-parameter tractable* (FPT) in k if it can be solved by an algorithm of runtime $f(k) \cdot n^{O(1)}$.
  (f: some computable function, n: input size)

Parametrized Complexity

- A problem is *fixed-parameter tractable* (FPT) in k if it can be solved by an algorithm of runtime $f(k) \cdot n^{O(1)}$.
  (f: some computable function, n: input size)

- A problem is in XP in k if it can be solved by an algorithm of runtime $n^{f(k)}$.

Parametrized Complexity

- A problem is *fixed-parameter tractable* (FPT) in k if it can be solved by an algorithm of runtime $f(k) \cdot n^{O(1)}$.
  (f: some computable function, n: input size)

- A problem is in XP in k if it can be solved by an algorithm of runtime $n^{f(k)}$.

- W[1]-hard problems are unlikely to be FPT.

Our Contribution

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

$P||C_{\max}$: NP-complete

$P|HM|C_{\max}$: poly. time

k : # job types

Our Contribution

So, what if we don't use identical machines?

P: identical machines

Q: related machines

R: unrelated machines

HM: high multipl. jobs

$C_{\max}$: min. makespan

$P||C_{\max}$: NP-complete

$P|HM|C_{\max}$: poly. time

k: # job types

Our Contribution

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

$P||C_{\max}$: NP-complete

$P|HM|C_{\max}$: poly. time

k : # job types

Our Contribution

P: identical machines
Q: related machines
R: unrelated machines
HM: high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k: # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

Cutting Stock is NP-hard already with 8 item types.

Our Contribution

P : identical machines
Q : related machines
R : unrelated machines
HM : high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k : # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

Cutting Stock is NP-hard already with 8 item types.

$Q||C_{\max}$ is W[1]-hard parameterized by k even for unary input.

Our Contribution

P: identical machines
Q: related machines
R: unrelated machines
HM: high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k: # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

Cutting Stock is NP-hard already with 8 item types.

$Q||C_{\max}$ is W[1]-hard parameterized by k even for unary input.

$P||C_{\max}$ is FPT parameterized by k.

Our Contribution

P: identical machines
Q: related machines
R: unrelated machines
HM: high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k: # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

Cutting Stock is NP-hard already with 8 item types.

$Q||C_{\max}$ is W[1]-hard parameterized by k even for unary input.

$P||C_{\max}$ is FPT parameterized by k.

$\{R, Q\}||C_{\max}$ is in XP parameterized by k, and
$\{R, Q\}|HM|C_{\max}$ is in XP param. by k if job sizes given unary.

Our Contribution

Agenda

P: identical machines
Q: related machines
R: unrelated machines
HM: high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k: # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

Cutting Stock is NP-hard already with 8 item types.

$Q||C_{\max}$ is W[1]-hard parameterized by k even for unary input.

$P||C_{\max}$ is FPT parameterized by k.

$\{R, Q\}||C_{\max}$ is in XP parameterized by k, and
$\{R, Q\}|HM|C_{\max}$ is in XP param. by k if job sizes given unary.

Our Contribution

Agenda

P: identical machines
Q: related machines
R: unrelated machines
HM: high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k: # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

1. hardness reduction

Cutting Stock is NP-hard already with 8 item types.

$Q||C_{\max}$ is W[1]-hard parameterized by k even for unary input.

$P||C_{\max}$ is FPT parameterized by k.

$\{R, Q\}||C_{\max}$ is in XP parameterized by k, and
$\{R, Q\}|HM|C_{\max}$ is in XP param. by k if job sizes given unary.

Our Contribution

Agenda

P: identical machines
Q: related machines
R: unrelated machines
HM: high multipl. jobs
$C_{\max}$: min. makespan
$P||C_{\max}$: NP-complete
$P|HM|C_{\max}$: poly. time
k: # job types

So, what if we don't use identical machines?

$Q|HM|C_{\max}$ is NP-hard (already for $k = 6$).

Cutting Stock is NP-hard already with 8 item types.

1. hardness reduction

$Q||C_{\max}$ is W[1]-hard parameterized by k even for unary input.

$P||C_{\max}$ is FPT parameterized by k.

2. obtaining FPT

$\{R, Q\}||C_{\max}$ is in XP parameterized by k, and
$\{R, Q\}|HM|C_{\max}$ is in XP param. by k if job sizes given unary.

Hardness Reduction: Overview

BIN PACKING (tight instances)
[known to be NP-complete]

Hardness Reduction: Overview

Bin Packing (tight instances)
[known to be NP-complete]

$\Big\downarrow$ reduces to

Balanced Bin Packing
(tight instances)

Hardness Reduction: Overview

BIN PACKING (tight instances)
[known to be NP-complete]

$\Big\downarrow$ reduces to

BALANCED BIN PACKING
(tight instances)

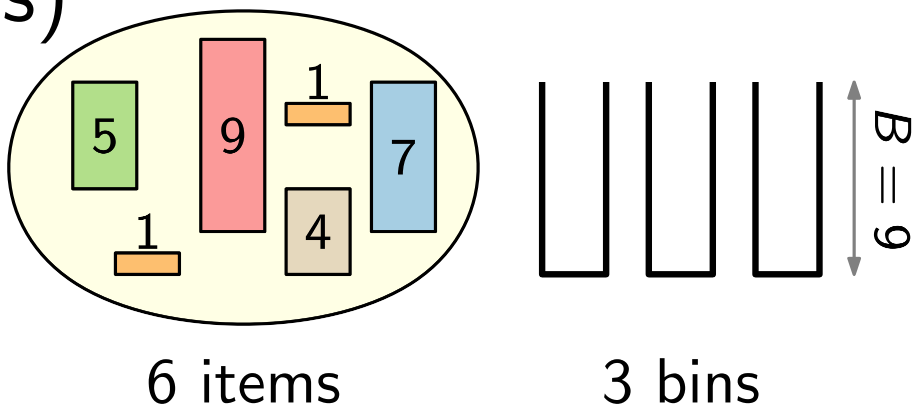
$\Big\downarrow$ reduces to

$Q|HM|C_{\max}$

Hardness Reduction: Overview

BIN PACKING (tight instances)

[known to be NP-complete]



$\Big\downarrow$ reduces to

BALANCED BIN PACKING
(tight instances)

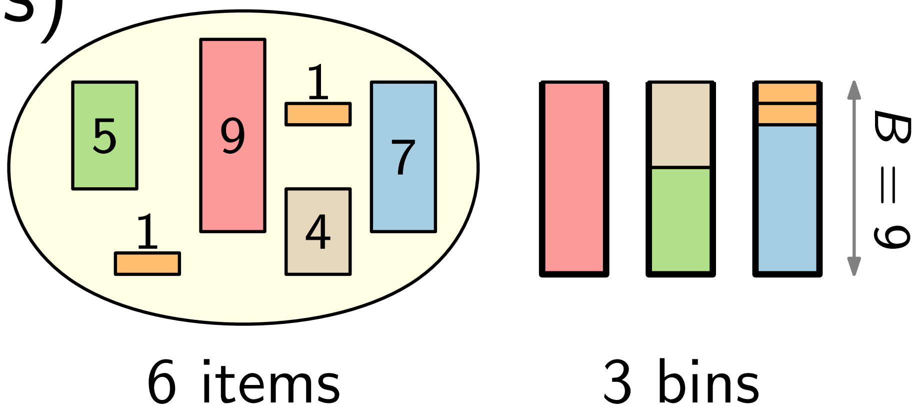
$\Big\downarrow$ reduces to

$Q|HM|C_{\max}$

Hardness Reduction: Overview

Bin Packing (tight instances)

[known to be NP-complete]



↓ reduces to

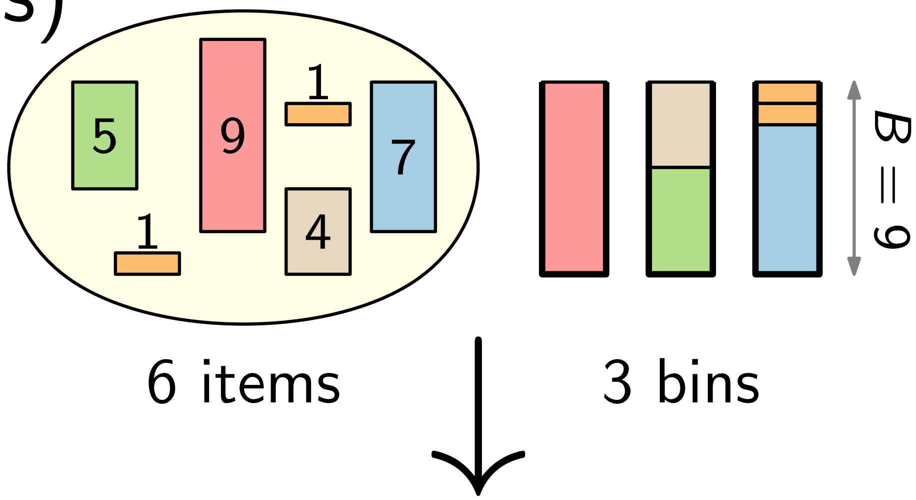
Balanced Bin Packing (tight instances)

↓ reduces to

$Q|HM|C_{\max}$

Hardness Reduction: Overview

BIN PACKING (tight instances)
[known to be NP-complete]



$\Downarrow$ reduces to

BALANCED BIN PACKING
(tight instances)

$\Downarrow$ reduces to

$Q|HM|C_{\max}$

Hardness Reduction: Overview

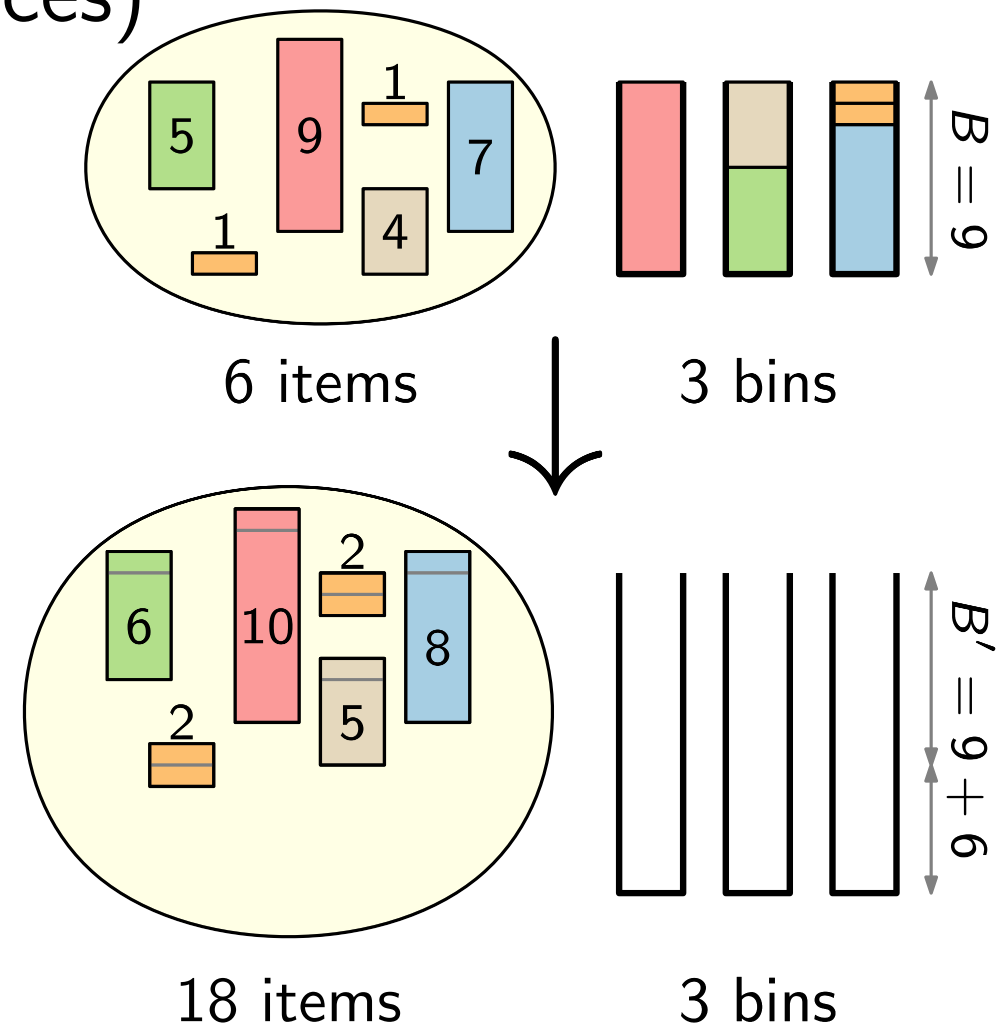
Bin Packing (tight instances)
[known to be NP-complete]

reduces to

Balanced Bin Packing (tight instances)

reduces to

$Q|HM|C_{\max}$



Hardness Reduction: Overview

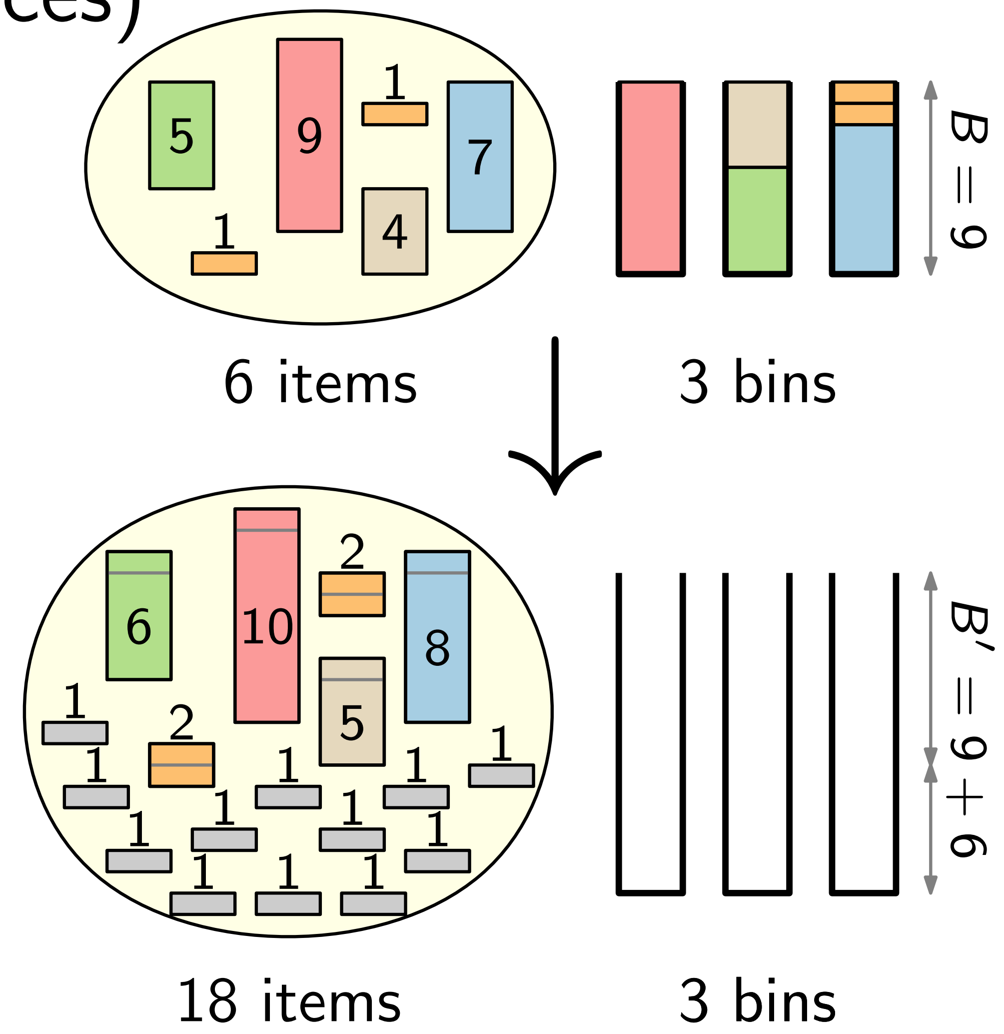
Bin Packing (tight instances)
[known to be NP-complete]

reduces to

Balanced Bin Packing (tight instances)

reduces to

$Q|HM|C_{\max}$



Hardness Reduction: Overview

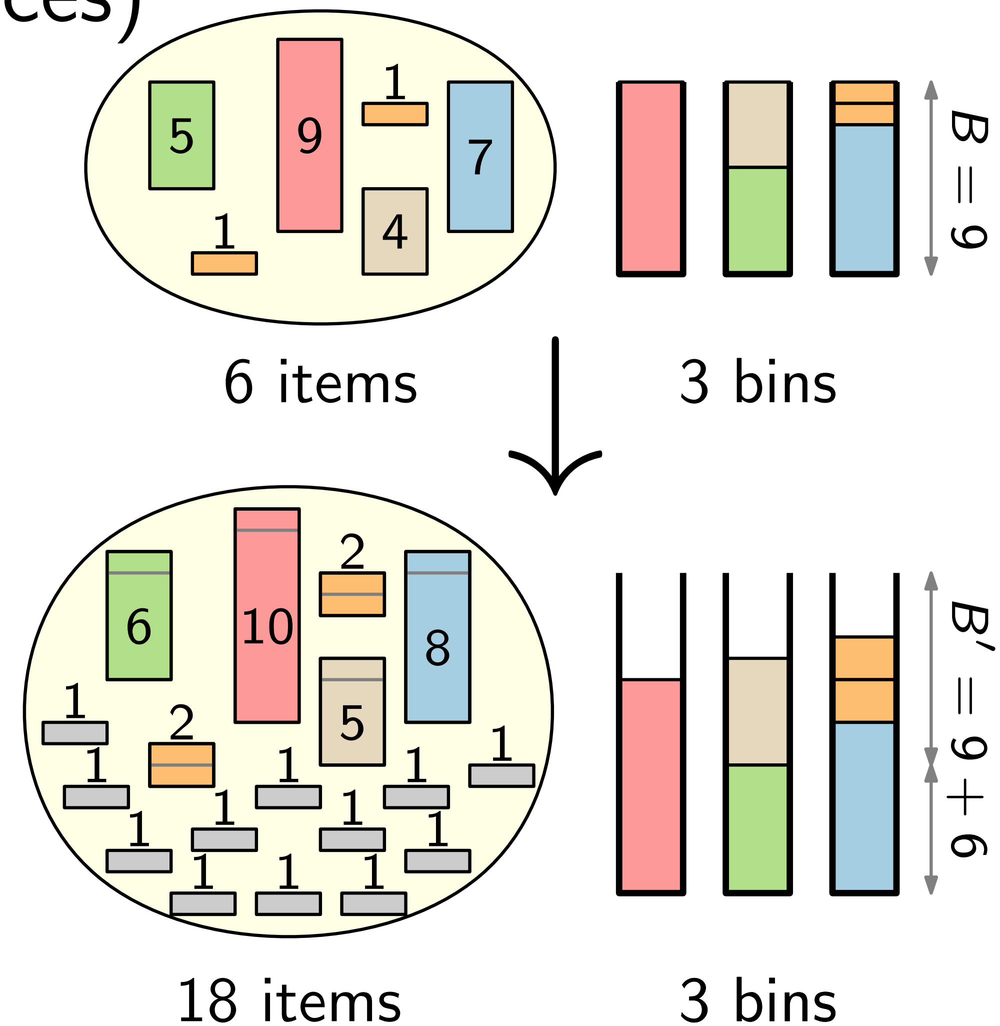
BIN PACKING (tight instances)
[known to be NP-complete]

reduces to

BALANCED BIN PACKING (tight instances)

reduces to

$Q|HM|C_{\max}$



Hardness Reduction: Overview

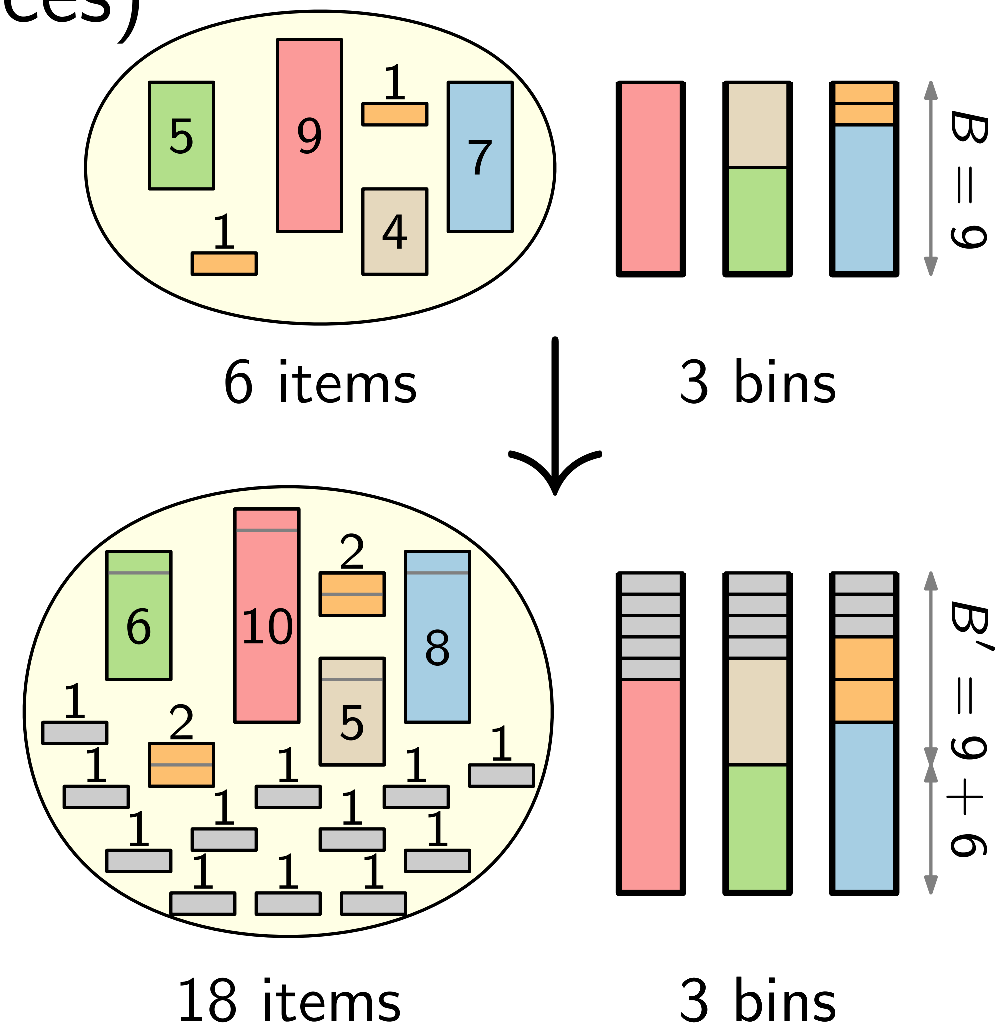
Bin Packing (tight instances)
[known to be NP-complete]

$\Big\downarrow$ reduces to

Balanced Bin Packing
(tight instances)

$\Big\downarrow$ reduces to

$Q|HM|C_{\max}$



Hardness Reduction: Overview

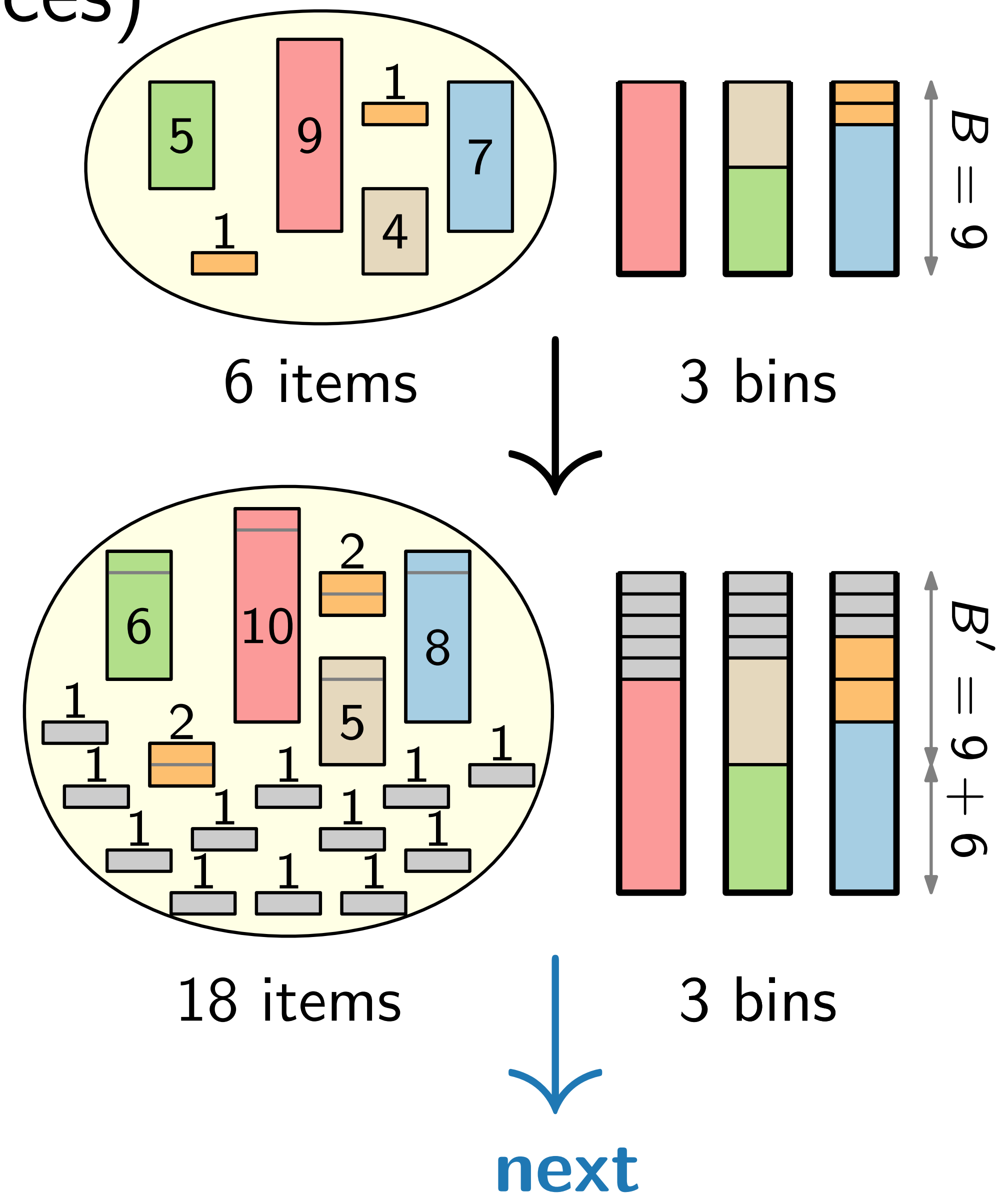
BIN PACKING (tight instances)
[known to be NP-complete]

reduces to

BALANCED BIN PACKING (tight instances)

reduces to

$Q|HM|C_{\max}$



Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:

Balanced Bin Packing $\rightarrow Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:
- for number a_i: use machine M_i ($\Rightarrow m$ machines)

Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:
- for number a_i: use machine M_i ($\Rightarrow$ m machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1

Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:
- for number a_i: use machine M_i ($\Rightarrow m$ machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1

$:= \{1, \ldots, k\}$

Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:

- for number a_i: use machine M_i ($\Rightarrow$ m machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1
- β_j on machine M_i $\Leftrightarrow$ a_i in the j-th bin

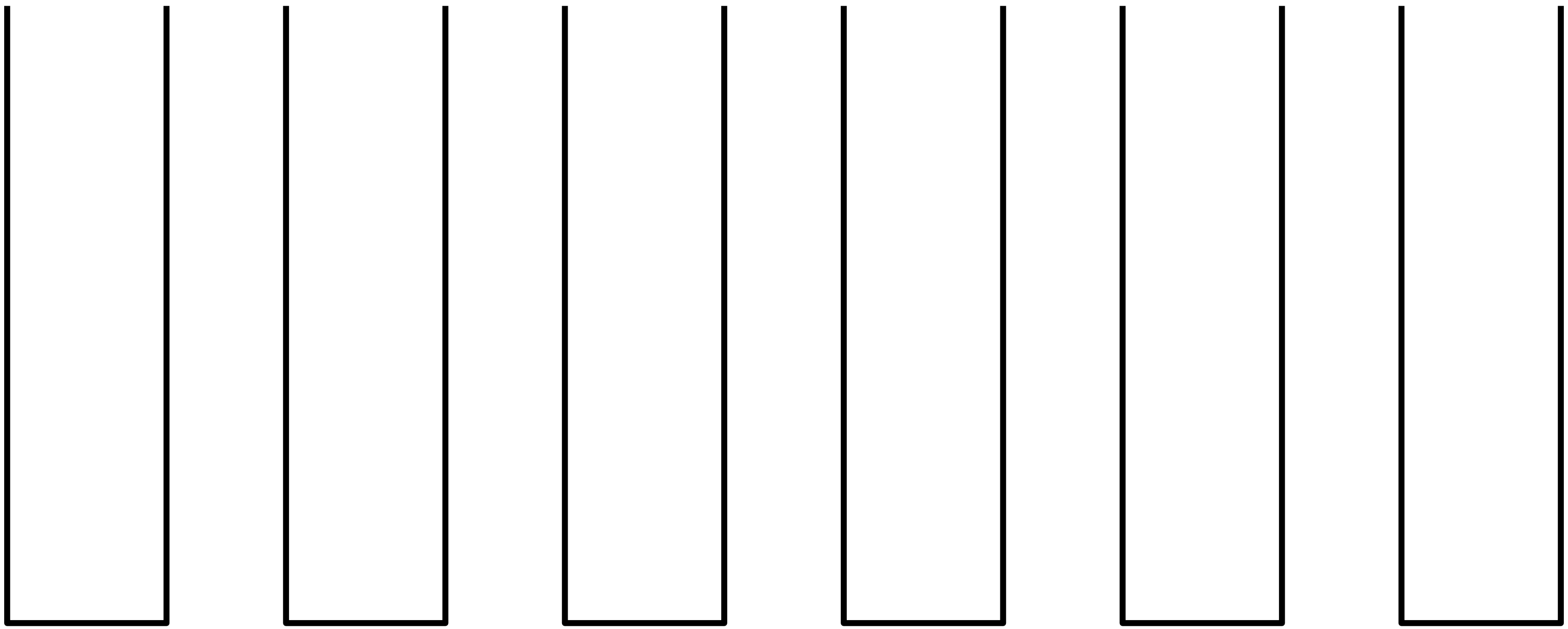
Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:

- for number a_i: use machine M_i ($\Rightarrow$ m machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1
- β_j on machine M_i $\Leftrightarrow$ a_i in the j-th bin

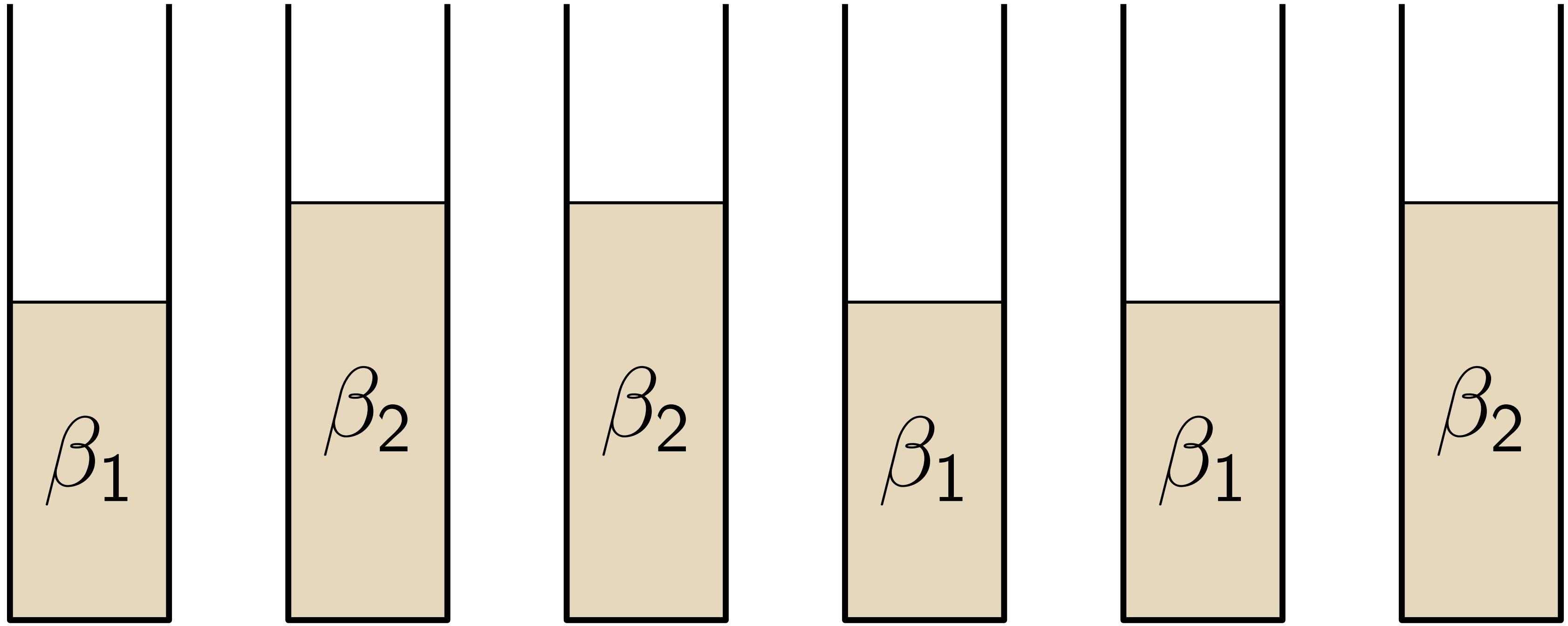


Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Idea:
- for number a_i: use machine M_i ($\Rightarrow m$ machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1
- β_j on machine $M_i \Leftrightarrow a_i$ in the j-th bin

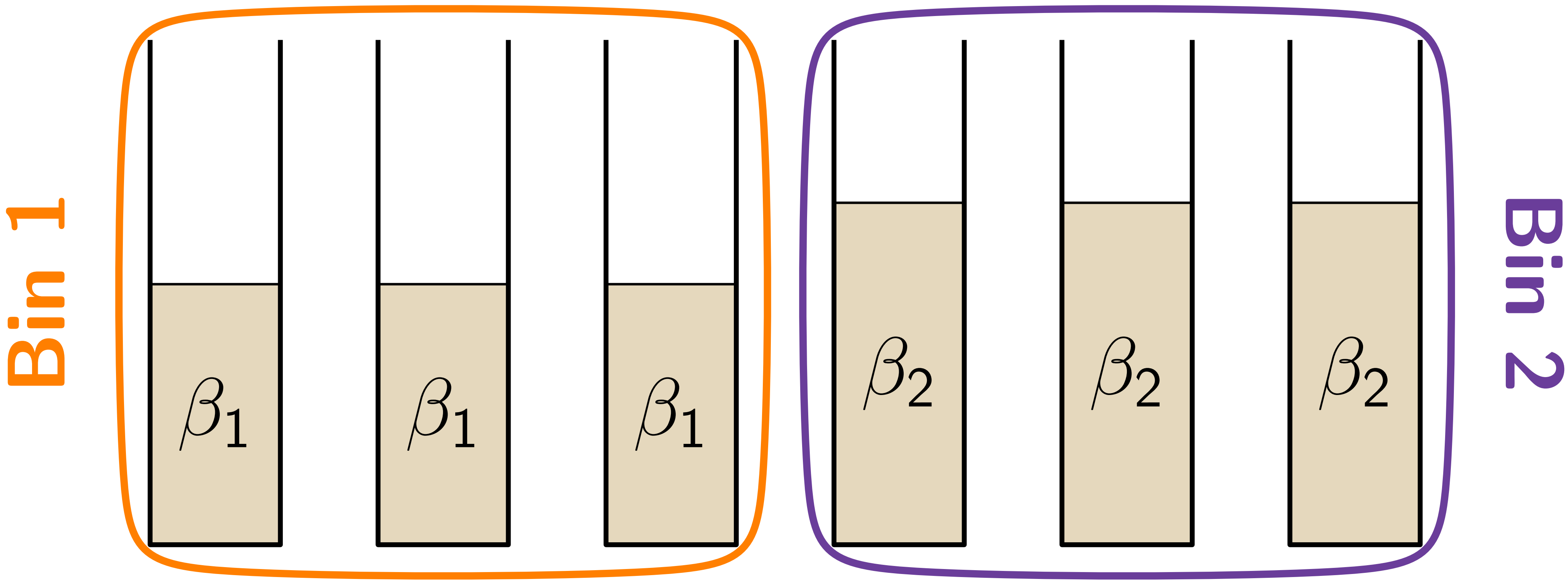


Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

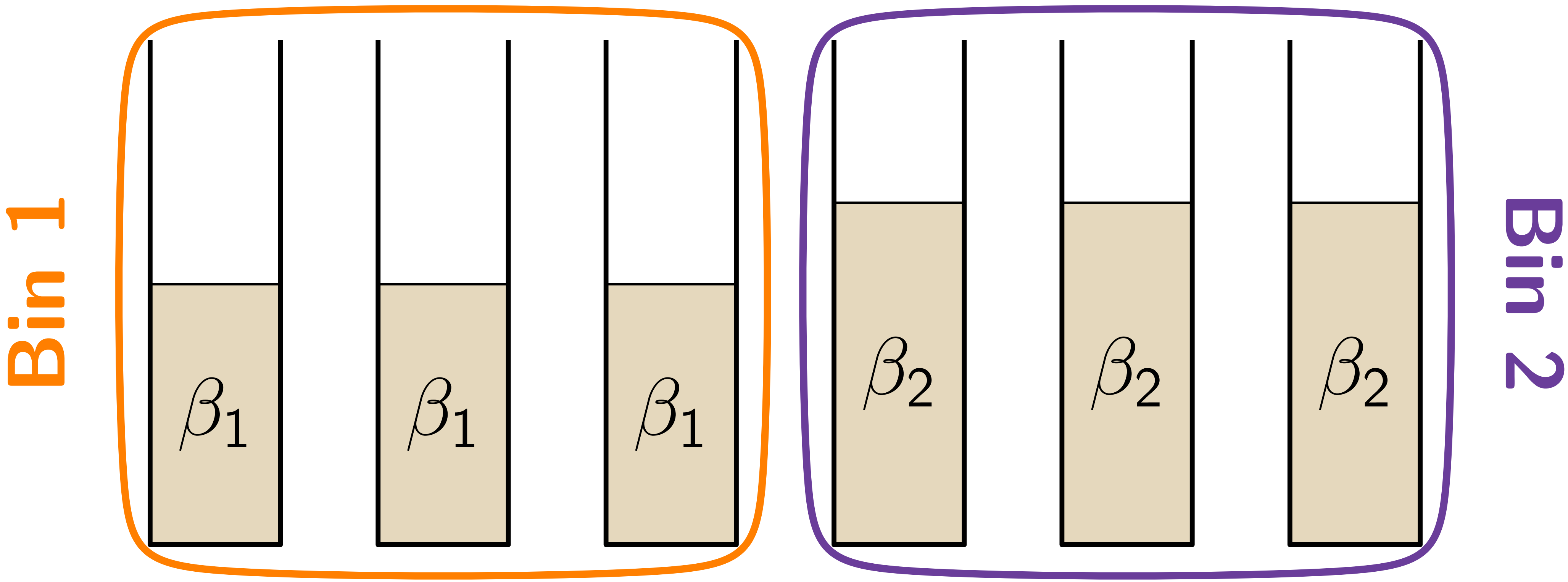
Idea:
- for number a_i: use machine M_i ($\Rightarrow$ m machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1
- β_j on machine M_i $\Leftrightarrow$ a_i in the j-th bin



Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

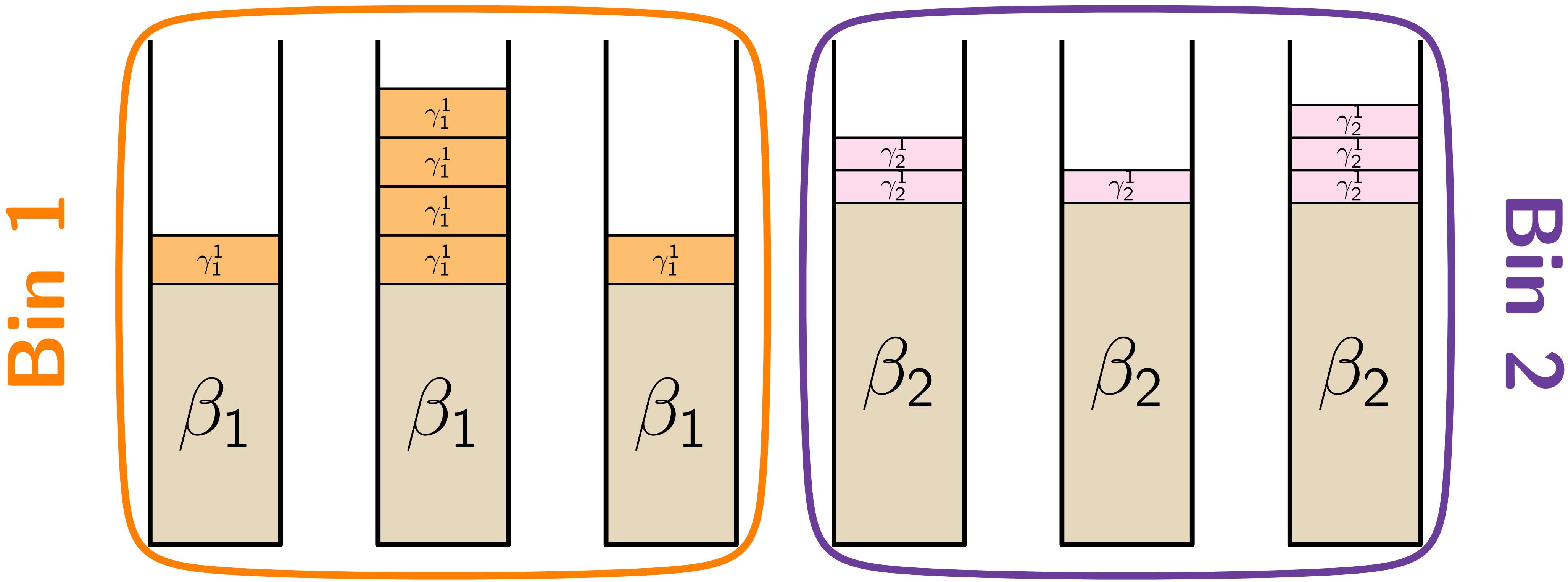
Idea:
- for number a_i: use machine M_i ($\Rightarrow m$ machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1
- β_j on machine $M_i \Leftrightarrow a_i$ in the j-th bin
- γ-type jobs enforce that each bin has the same sum of items



Balanced Bin Packing → $Q|HM|C_{max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

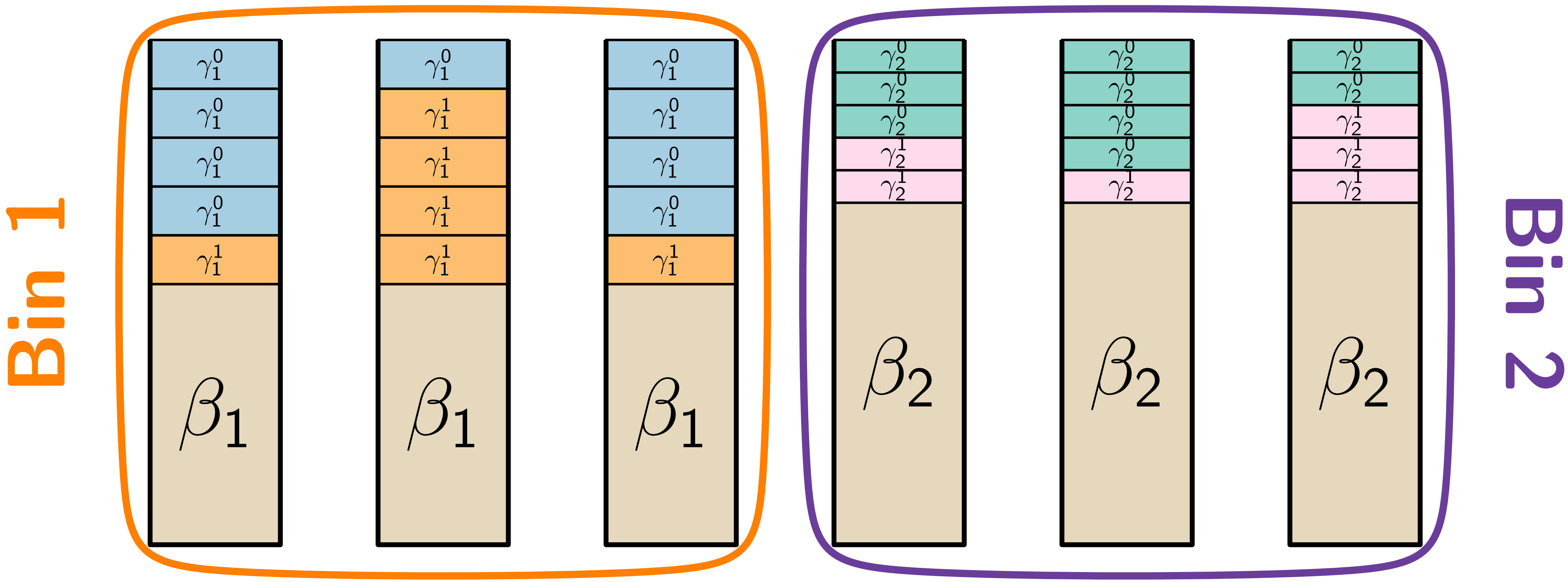
Idea:
- for number a_i: use machine M_i ($\Rightarrow$ m machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0, γ_j^1
- β_j on machine M_i $\Leftrightarrow$ a_i in the j-th bin
- γ-type jobs enforce that each bin has the same sum of items



Balanced Bin Packing $\rightarrow Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

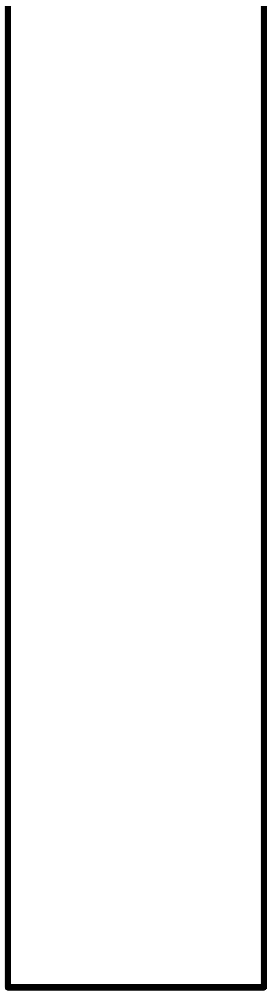
Idea:
- for number a_i: use machine M_i ($\Rightarrow$ m machines)
- for bin $j \in [k]$: use job types β_j, γ_j^0 , γ_j^1
- β_j on machine $M_i \Leftrightarrow a_i$ in the j-th bin
- γ-type jobs enforce that each bin has the same sum of items



Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

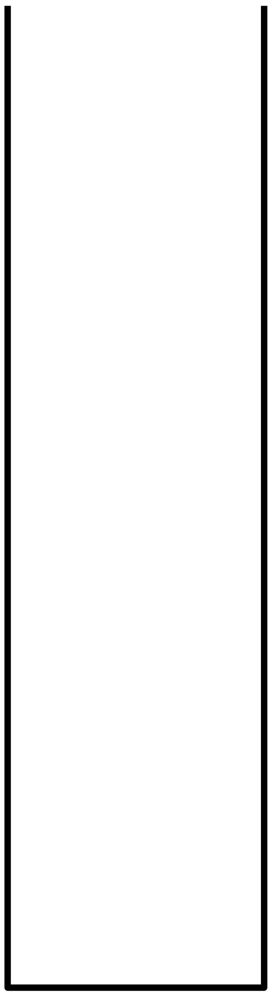


BALANCED BIN PACKING $\to Q|HM|C_{\max}$

BALANCED BIN PACKING instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.



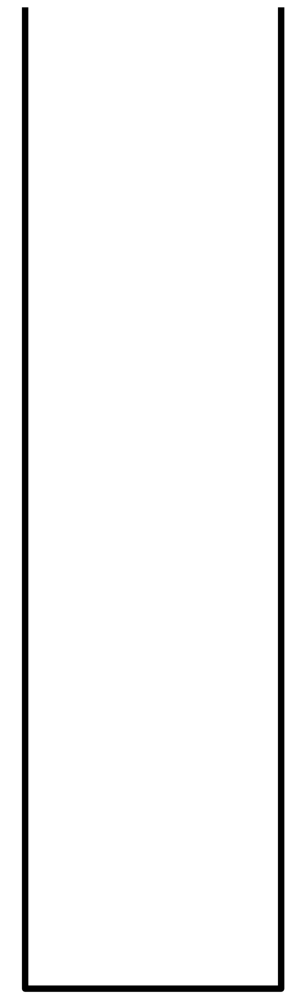
Machine M_i

Balanced Bin Packing $\rightarrow Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$



Machine M_i

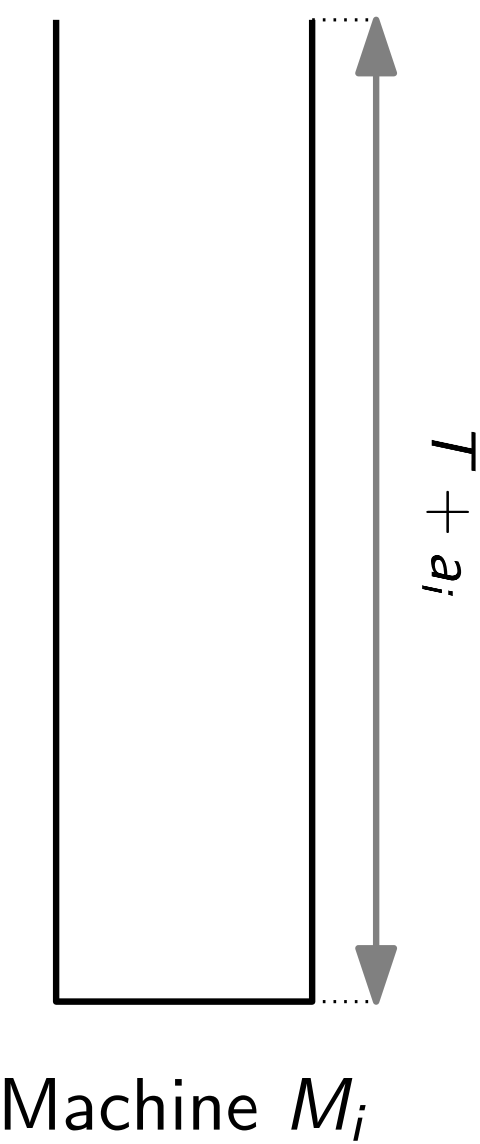
Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$



Balanced Bin Packing → $Q|HM|C_{\max}$

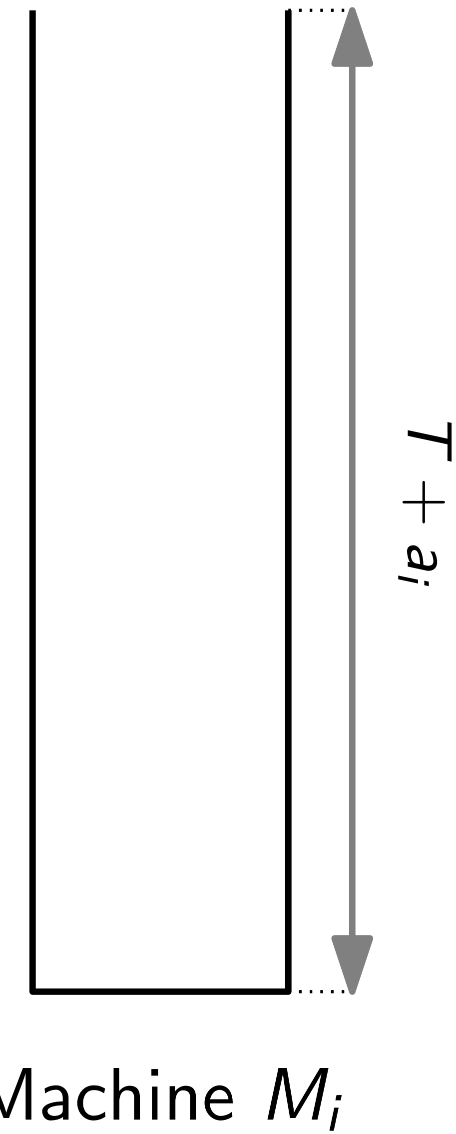
Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$

- β-type jobs are so large that we have ≤ 1 per machine.



Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

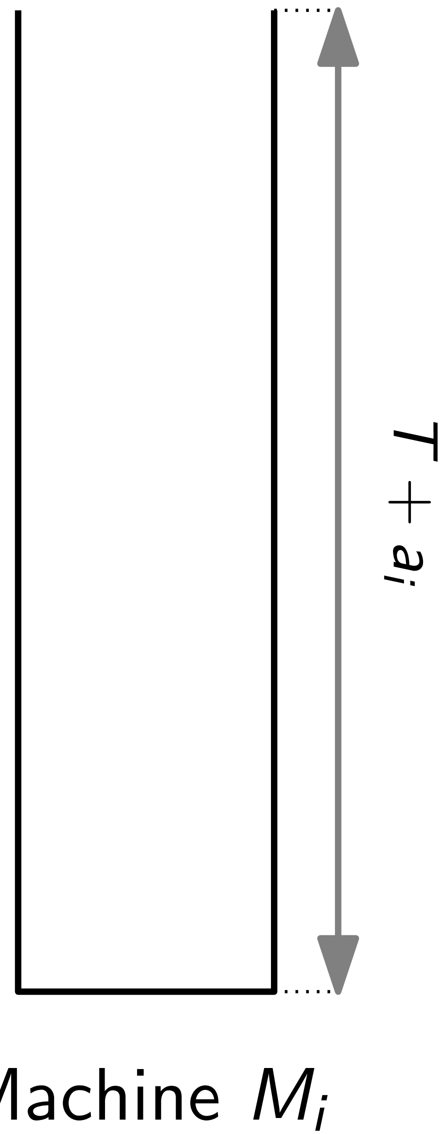
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$

- β-type jobs are so large that we have ≤ 1 per machine.

- We have m machines & m β-type jobs.



Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

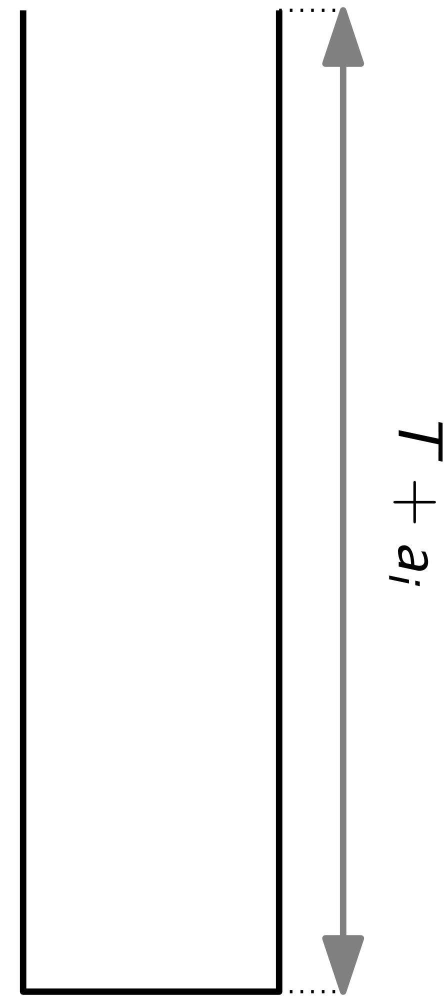
- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$

- β-type jobs are so large that we have ≤ 1 per machine.

- We have m machines & m β-type jobs.

$\Rightarrow$ One β-type job / machine



Machine M_i

Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:

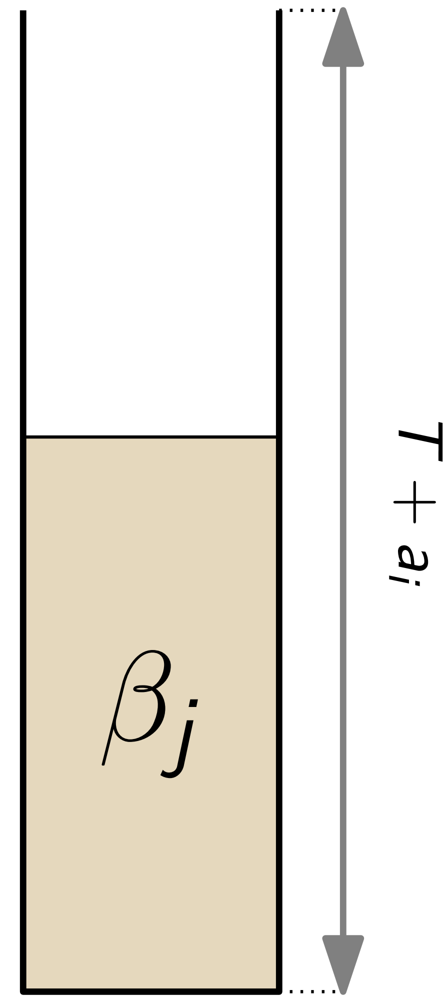
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$

- β-type jobs are so large that we have ≤ 1 per machine.
- We have m machines & m β-type jobs.

$\Rightarrow$ One β-type job / machine



Machine M_i

BALANCED BIN PACKING → $Q|HM|C_{\max}$

BALANCED BIN PACKING instance:
- multiset of numbers $\{a_1, \dots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

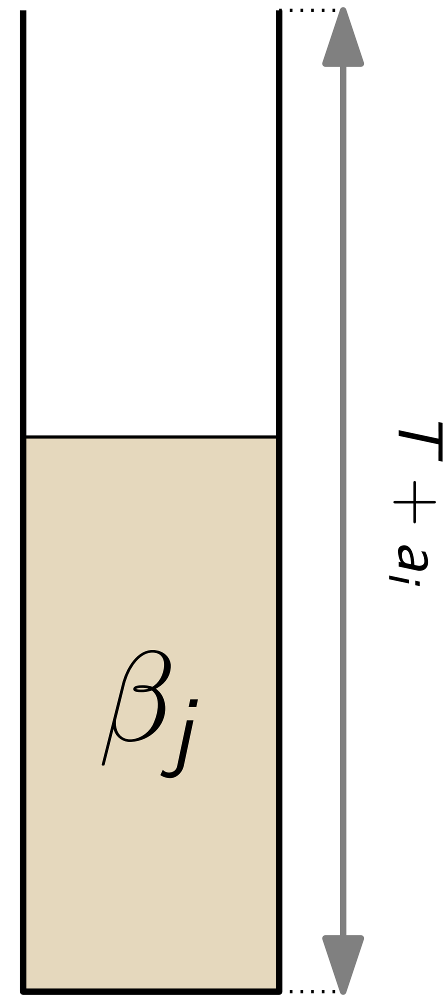
- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$

- β-type jobs are so large that we have ≤ 1 per machine.
- We have m machines & m β-type jobs.

$\Rightarrow$ One β-type job / machine

- β_j leaves space of $At_j + a_i$



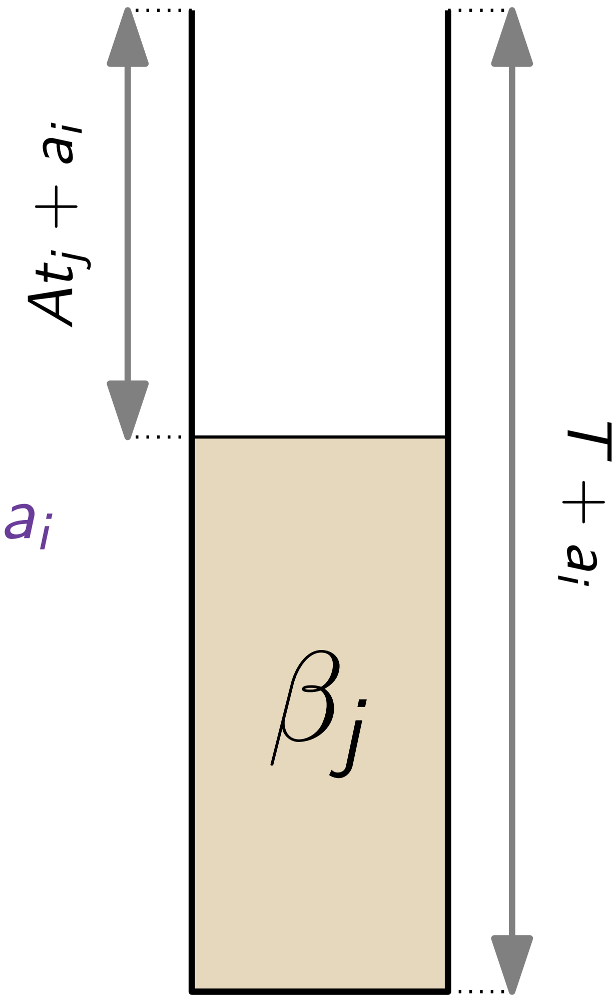
Machine M_i

BALANCED BIN PACKING $\to Q|HM|C_{\max}$

BALANCED BIN PACKING instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

- Speed of M_i is slightly faster than 1. Allows to process jobs of lengths $T + a_i$ in time T.

$\Rightarrow$ think of M_i has "capacity" $T + a_i$



Machine M_i

- β-type jobs are so large that we have ≤ 1 per machine.
- We have m machines & m β-type jobs.

$\Rightarrow$ One β-type job / machine

- β_j leaves space of $At_j + a_i$

Balanced Bin Packing → $Q|HM|C_{\text{max}}$

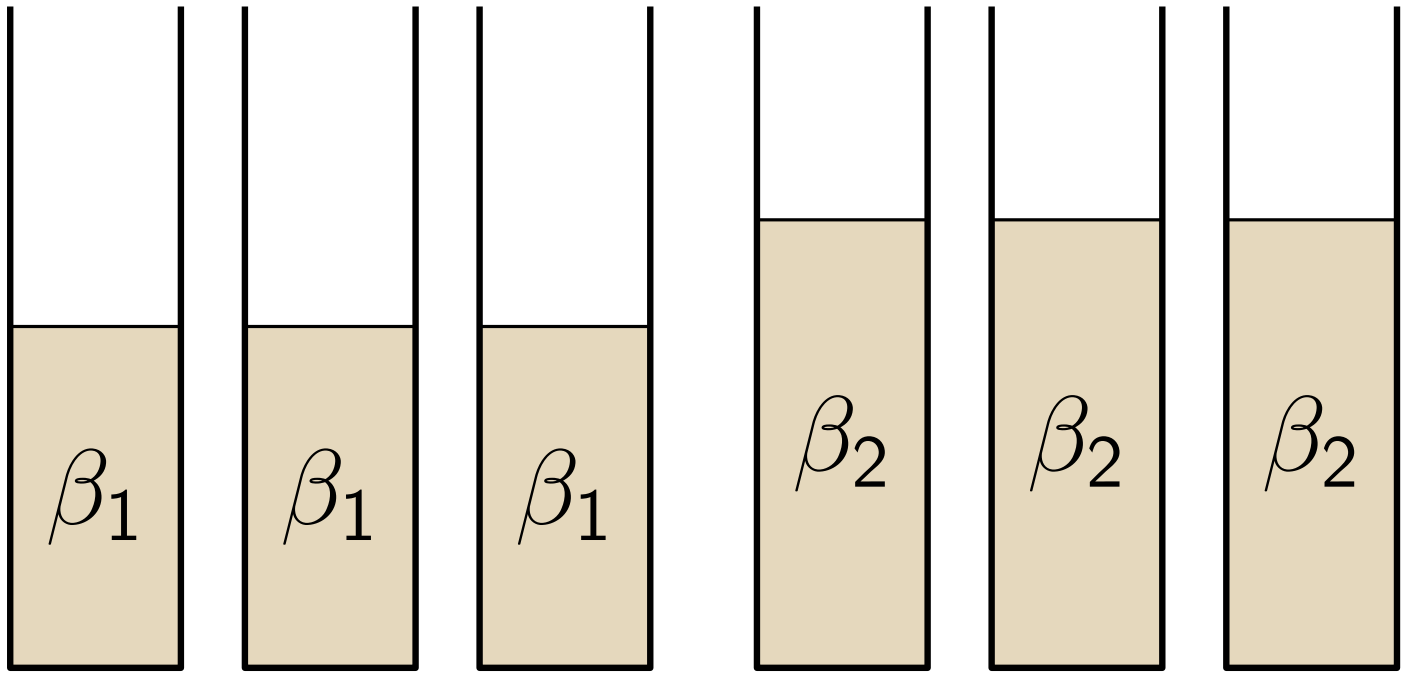
Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

Balanced Bin Packing $\rightarrow Q|HM|C_{\max}$

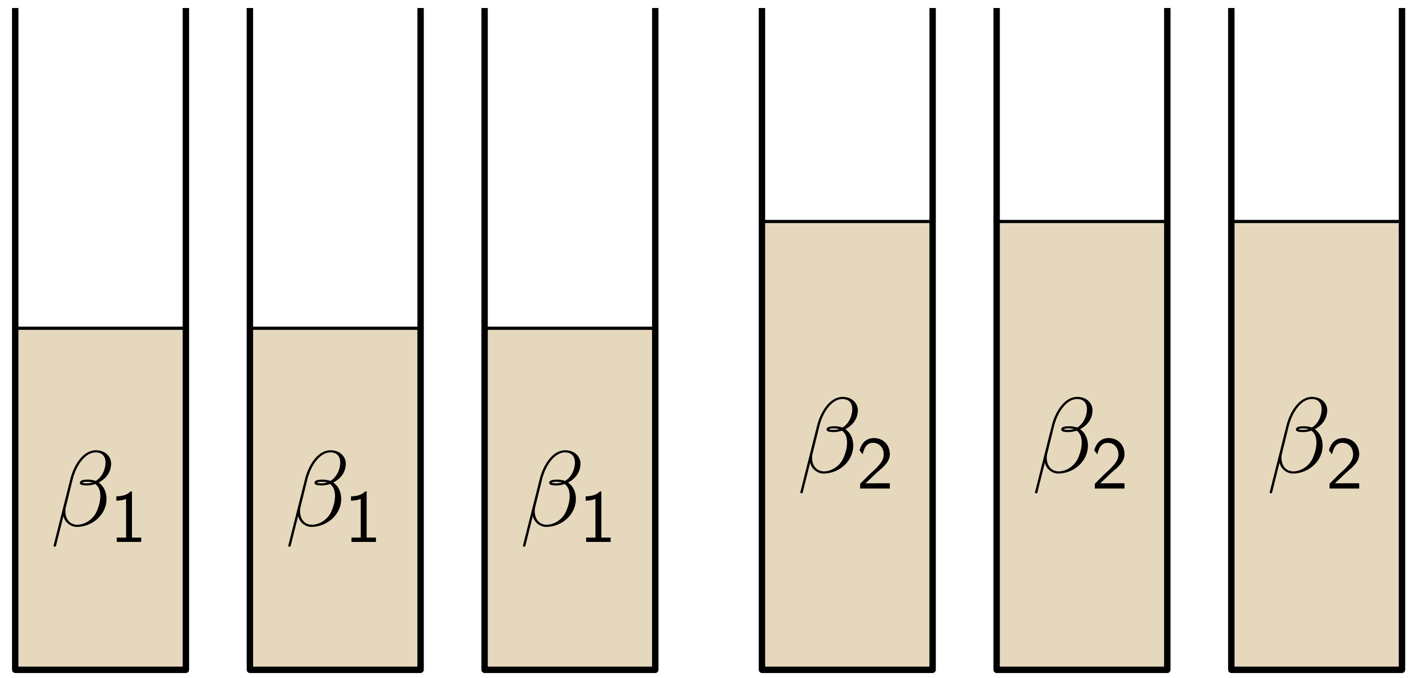
Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$



Balanced Bin Packing → $Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

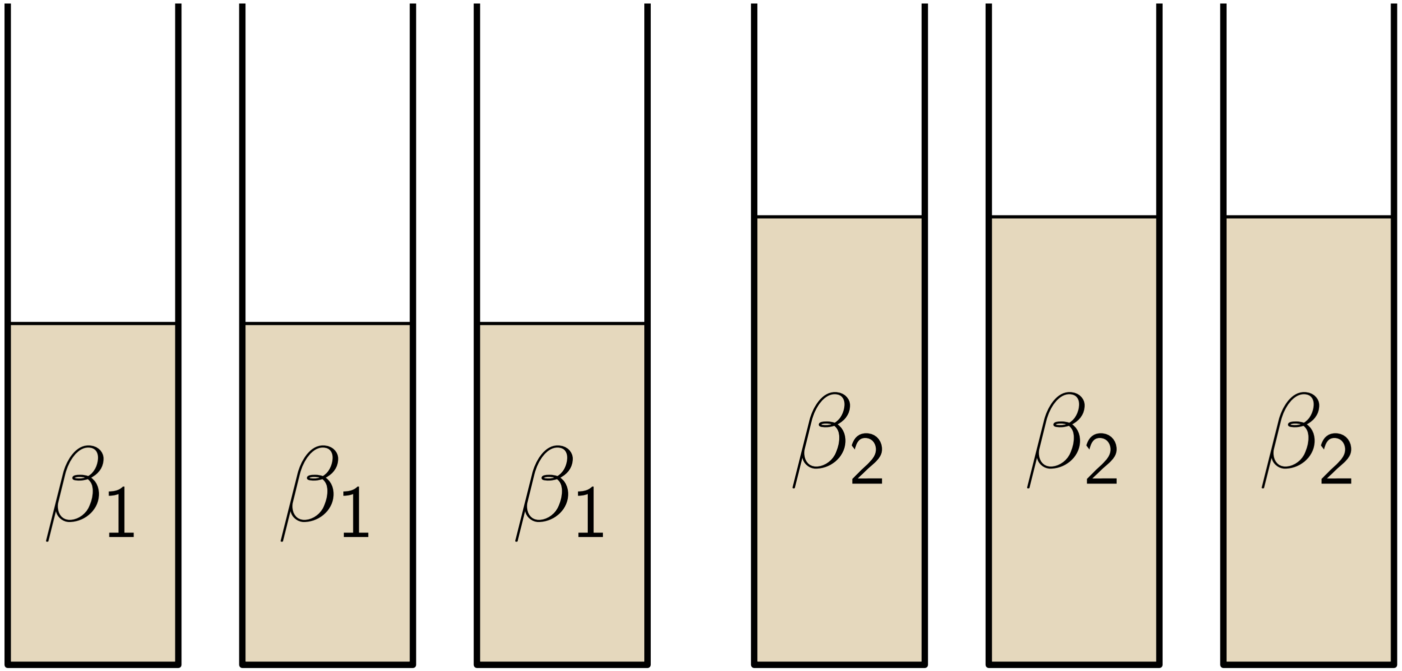


- We have mA γ-type jobs

Balanced Bin Packing → $Q|HM|C_{max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

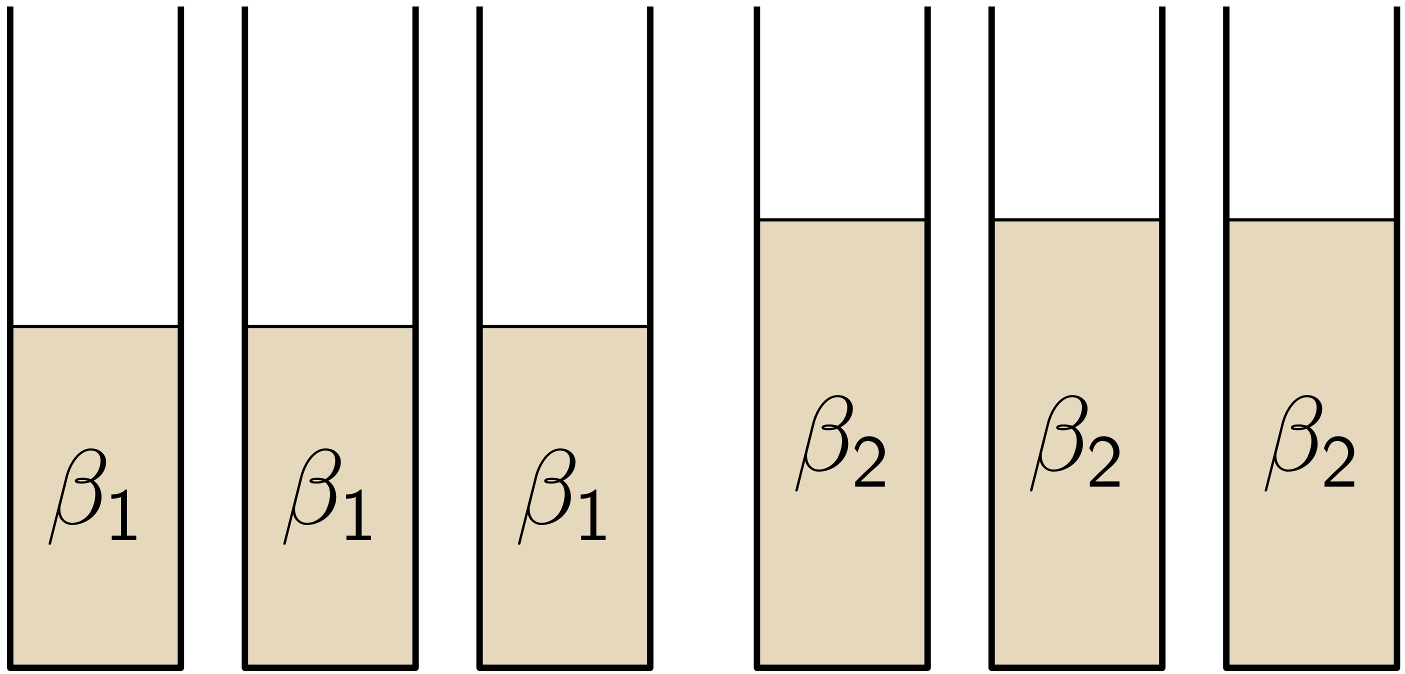


- We have mA γ-type jobs
- Any machine can execute $\leq A$ γ-type jobs

Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$



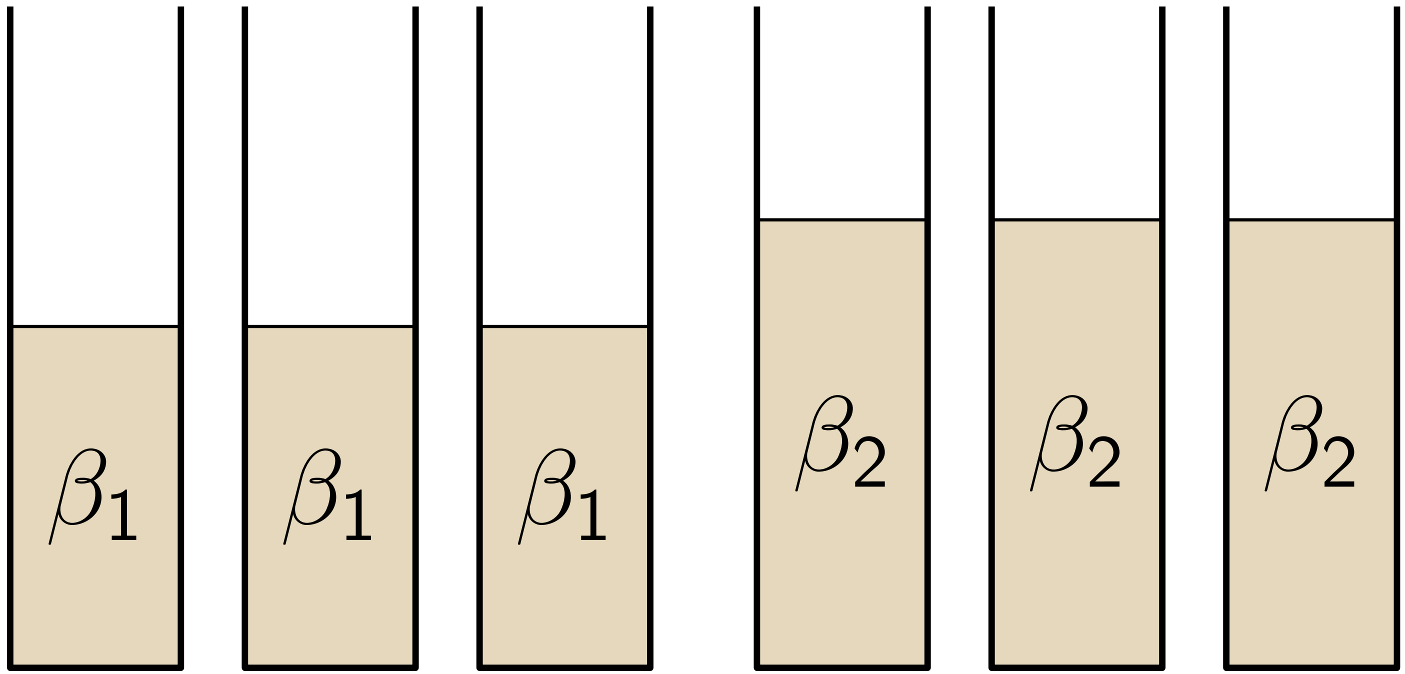
- We have mA γ-type jobs
- Any machine can execute $\leq A$ γ-type jobs

$\Rightarrow$ exactly A γ-type jobs / machine

Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$



- We have mA γ-type jobs

- Any machine can execute $\leq A$ γ-type jobs

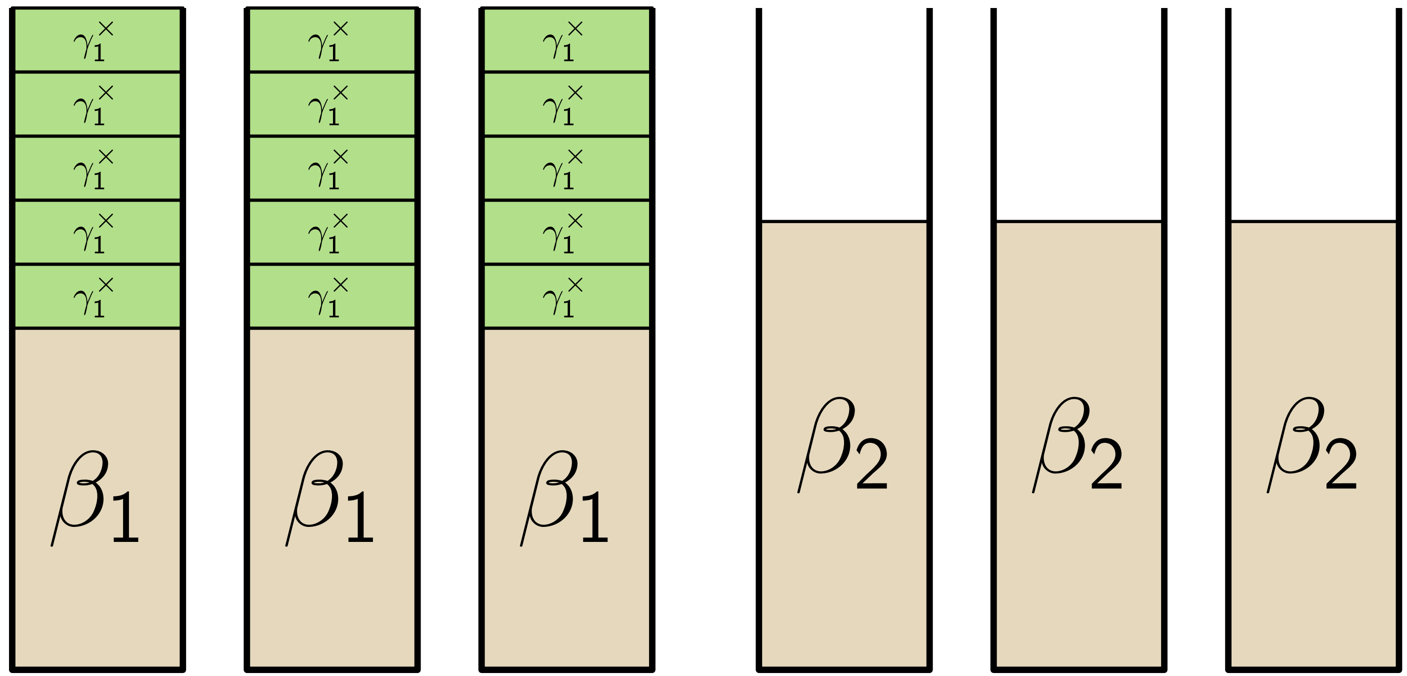
$\Rightarrow$ exactly A γ-type jobs / machine

- As $\gamma_1^\times > \gamma_2^\times$, only machines with β_1 can host A $\gamma_1^\times$ jobs

Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$



- We have mA γ-type jobs
- Any machine can execute $\leq A$ γ-type jobs

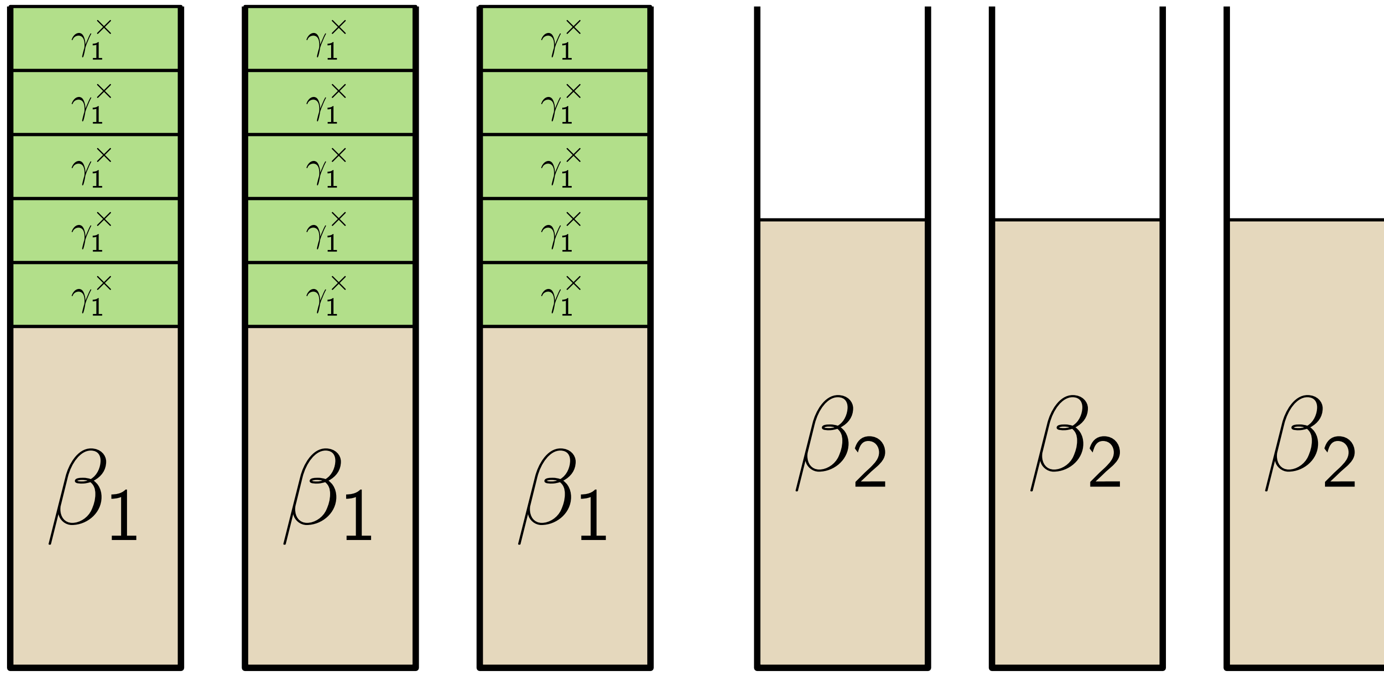
$\Rightarrow$ exactly A γ-type jobs / machine

- As $\gamma_1^\times > \gamma_2^\times$, only machines with β_1 can host A $\gamma_1^\times$ jobs

BALANCED BIN PACKING $\to Q|HM|C_{\max}$

BALANCED BIN PACKING instance:

- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$



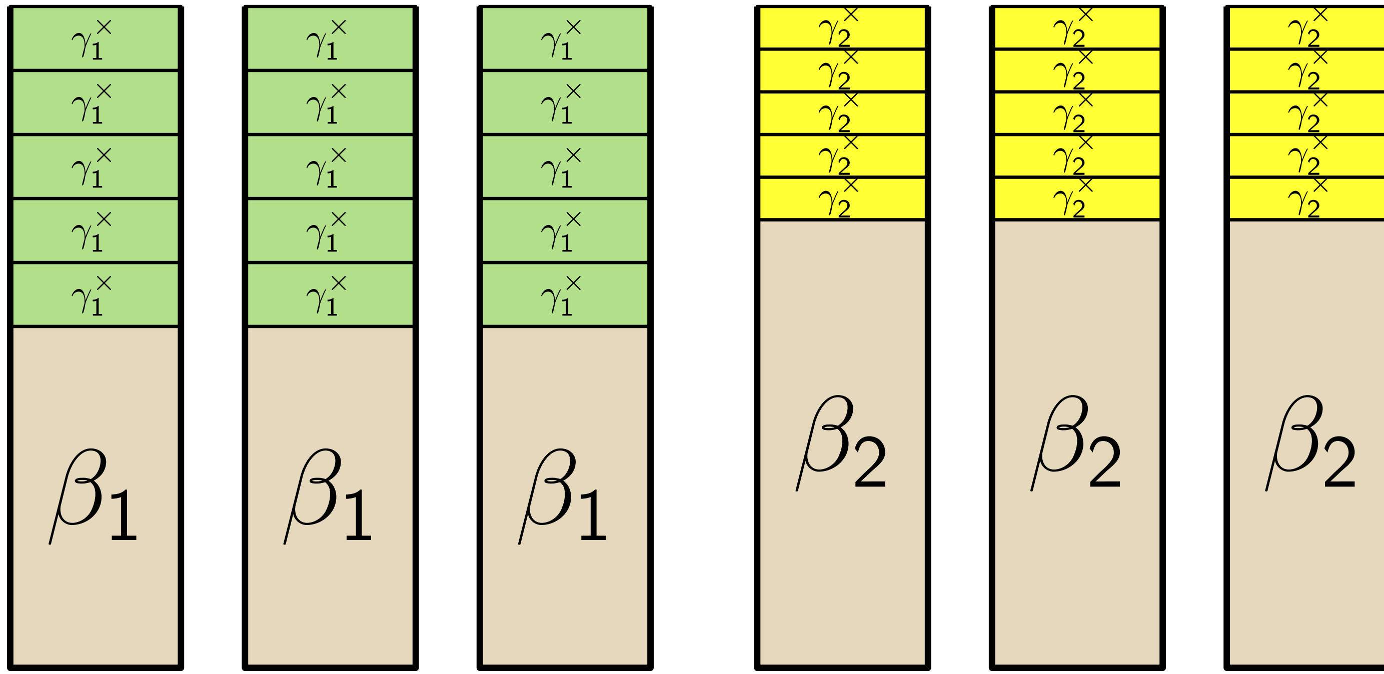
- We have mA γ-type jobs
- Any machine can execute $\le A$ γ-type jobs

$\Rightarrow$ exactly A γ-type jobs / machine

- As $\gamma_1^\times > \gamma_2^\times$, only machines with β_1 can host A $\gamma_1^\times$ jobs
- Only machines with β_2 remain for $\gamma_2^\times$, and so on

Balanced Bin Packing $\to Q|HM|C_{\max}$

Balanced Bin Packing instance:
- multiset of numbers $\{a_1, \ldots, a_m\}$
- k bins of size B
- $A := \sum_{i=1}^{m} a_i = kB$

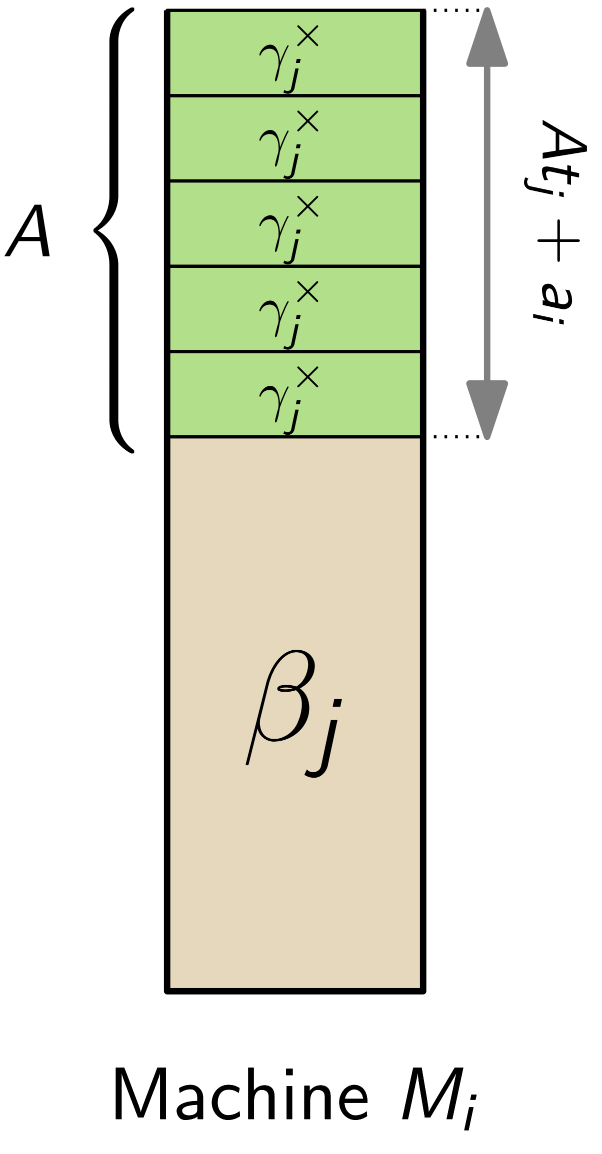


- We have mA γ-type jobs
- Any machine can execute $\leq A$ γ-type jobs

$\Rightarrow$ exactly A γ-type jobs / machine

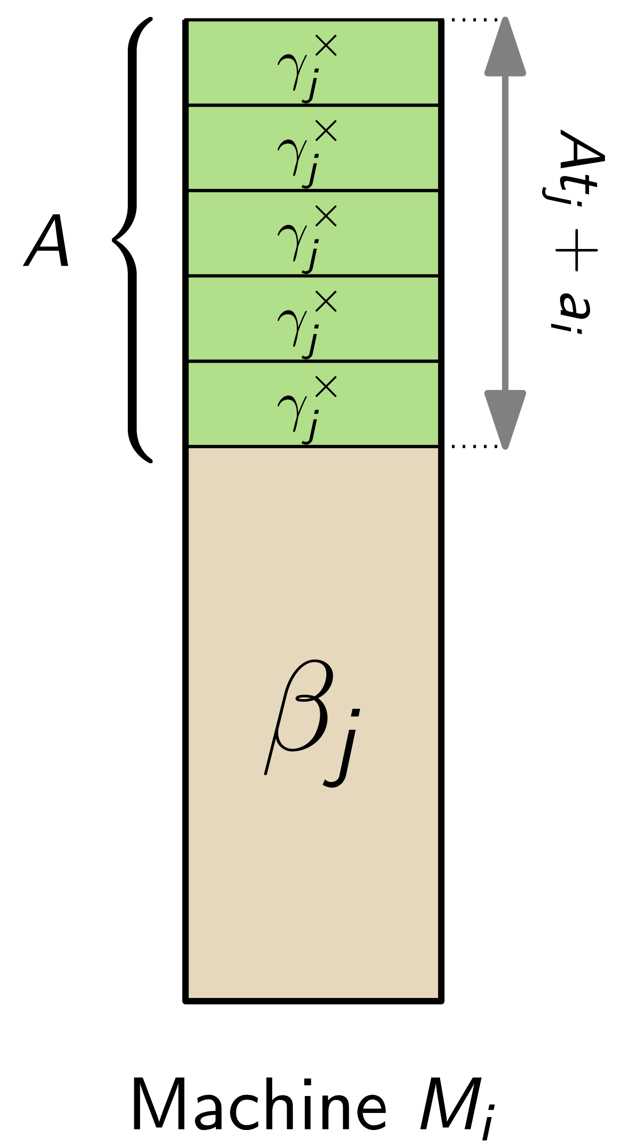
- As $\gamma_1^\times > \gamma_2^\times$, only machines with β_1 can host A $\gamma_1^\times$ jobs
- Only machines with β_2 remain for $\gamma_2^\times$, and so on

Balanced Bin Packing $\to Q|HM|C_{\max}$



Balanced Bin Packing $\to Q|HM|C_{\max}$

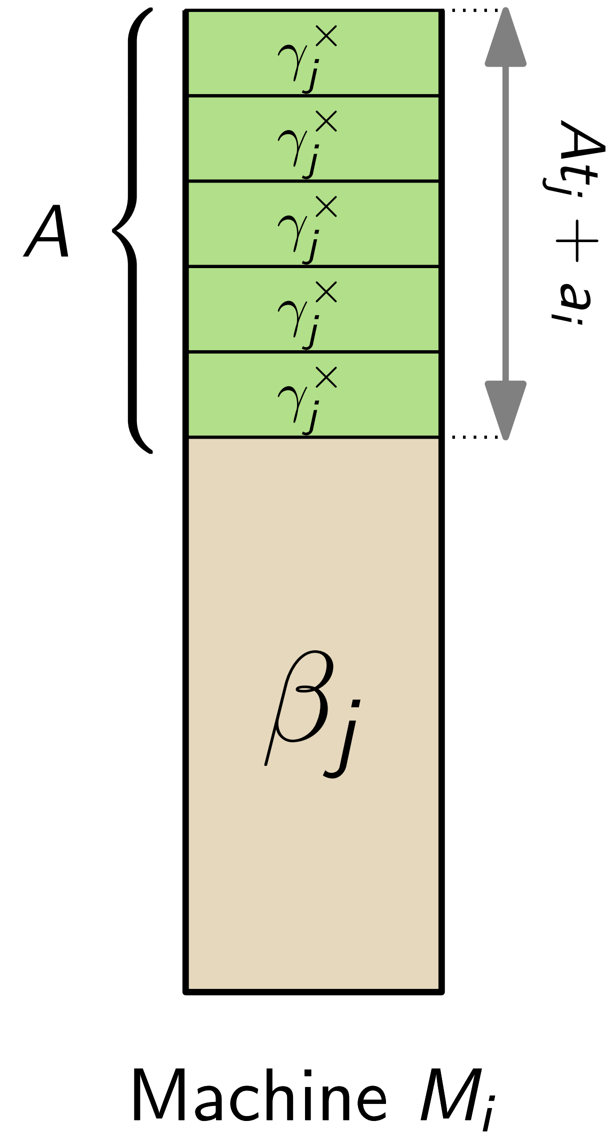
Distinguishing between γ_j^0 and γ_j^1:



Balanced Bin Packing $\to Q|HM|C_{\max}$

Distinguishing between γ_j^0 and γ_j^1:

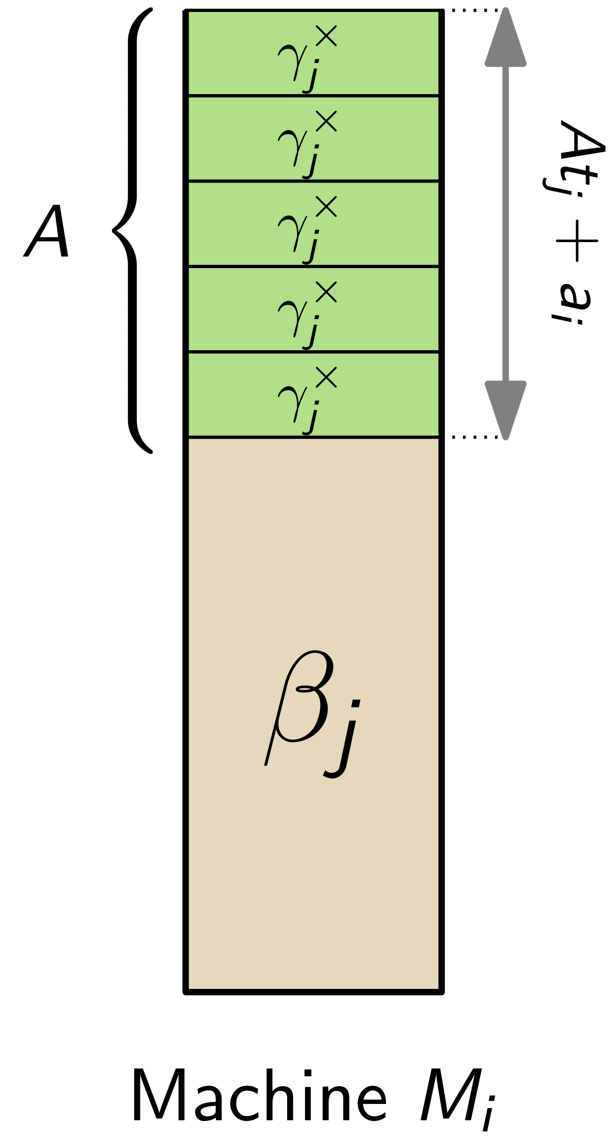
- $p_{\gamma_j^0} = t_j$



Balanced Bin Packing → $Q|HM|C_{\max}$

Distinguishing between γ_j^0 and γ_j^1:

- $p_{\gamma_j^0} = t_j$
- $p_{\gamma_j^1} = t_j + 1$

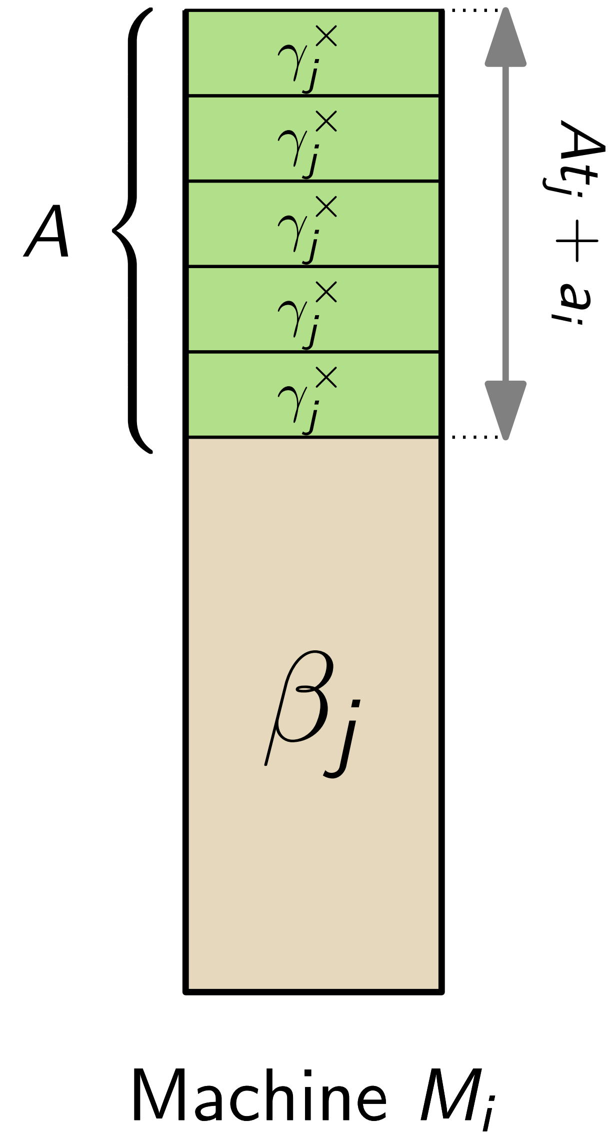


Balanced Bin Packing $\rightarrow Q|HM|C_{\max}$

Distinguishing between γ_j^0 and γ_j^1:

- $p_{\gamma_j^0} = t_j$
- $p_{\gamma_j^1} = t_j + 1$

$\Rightarrow \; \leq a_i$ jobs of type γ_j^1 per M_i



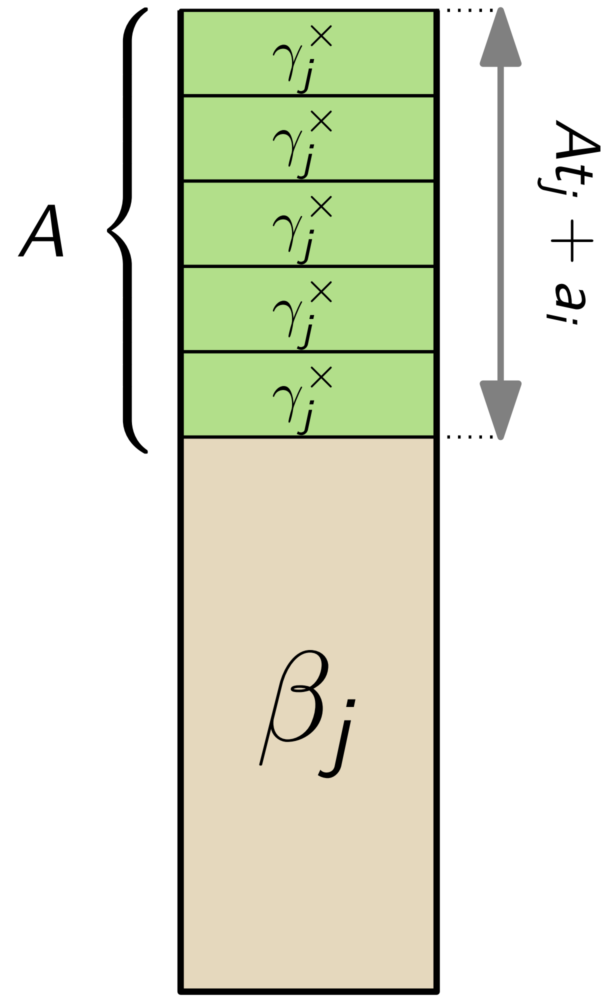
Balanced Bin Packing $\to Q|HM|C_{\max}$

Distinguishing between γ_j^0 and γ_j^1:

- $p_{\gamma_j^0} = t_j$
- $p_{\gamma_j^1} = t_j + 1$

$\Rightarrow \ \leq a_i$ jobs of type γ_j^1 per M_i

$\quad\ \ \geq A - a_i$ jobs of type γ_j^0 per M_i



Machine M_i

Balanced Bin Packing $\to Q|HM|C_{\max}$

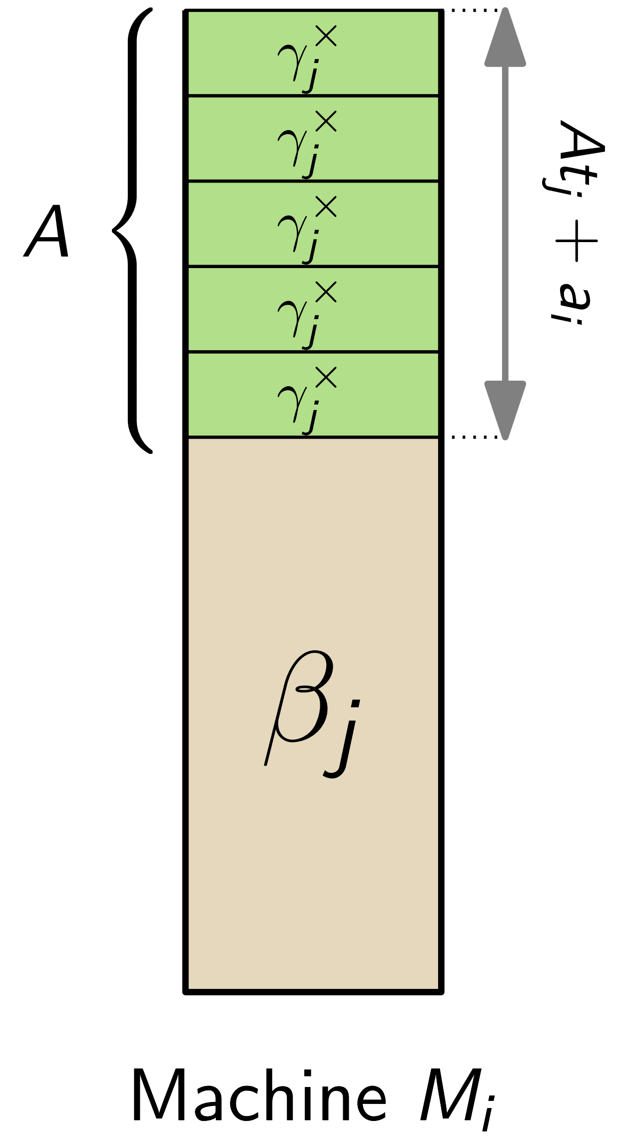
Distinguishing between γ_j^0 and γ_j^1:

- $p_{\gamma_j^0} = t_j$
- $p_{\gamma_j^1} = t_j + 1$

$\Rightarrow \leq a_i$ jobs of type γ_j^1 per M_i

$\geq A - a_i$ jobs of type γ_j^0 per M_i

- We have $A = \sum_{i=0}^{m} a_i$ jobs of type $\gamma_\times^1$ in total.



Balanced Bin Packing $\to Q|HM|C_{\max}$

Distinguishing between γ_j^0 and γ_j^1:

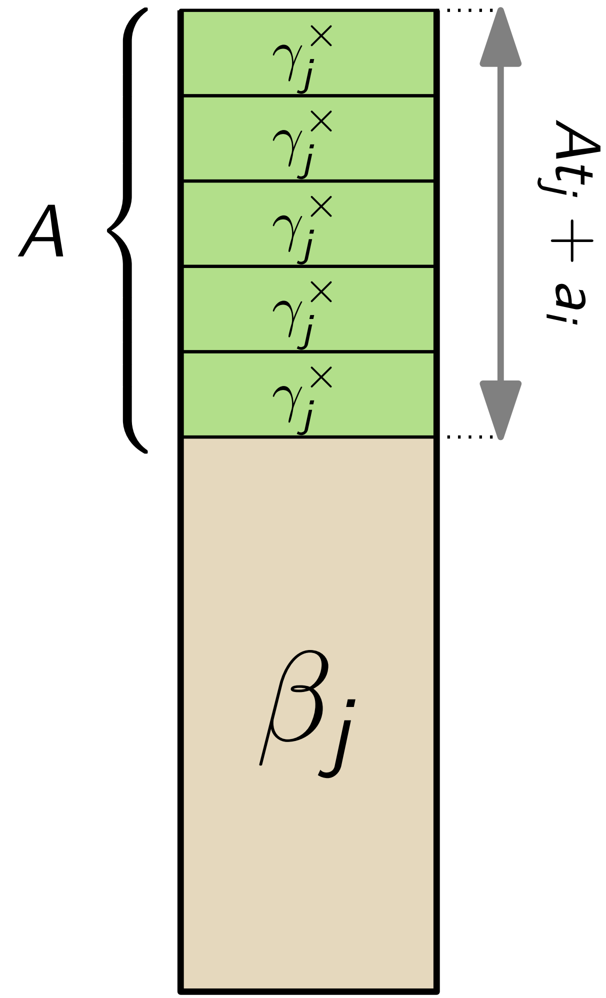
- $p_{\gamma_j^0} = t_j$
- $p_{\gamma_j^1} = t_j + 1$

$\Rightarrow \leq a_i$ jobs of type γ_j^1 per M_i

$\geq A - a_i$ jobs of type γ_j^0 per M_i

- We have $A = \sum_{i=0}^{m} a_i$ jobs of type $\gamma_\times^1$ in total.

$\Rightarrow$ M_i has exactly a_i jobs of type γ_j^1



Machine M_i

Balanced Bin Packing $\to Q|HM|C_{\max}$

Distinguishing between γ_j^0 and γ_j^1:

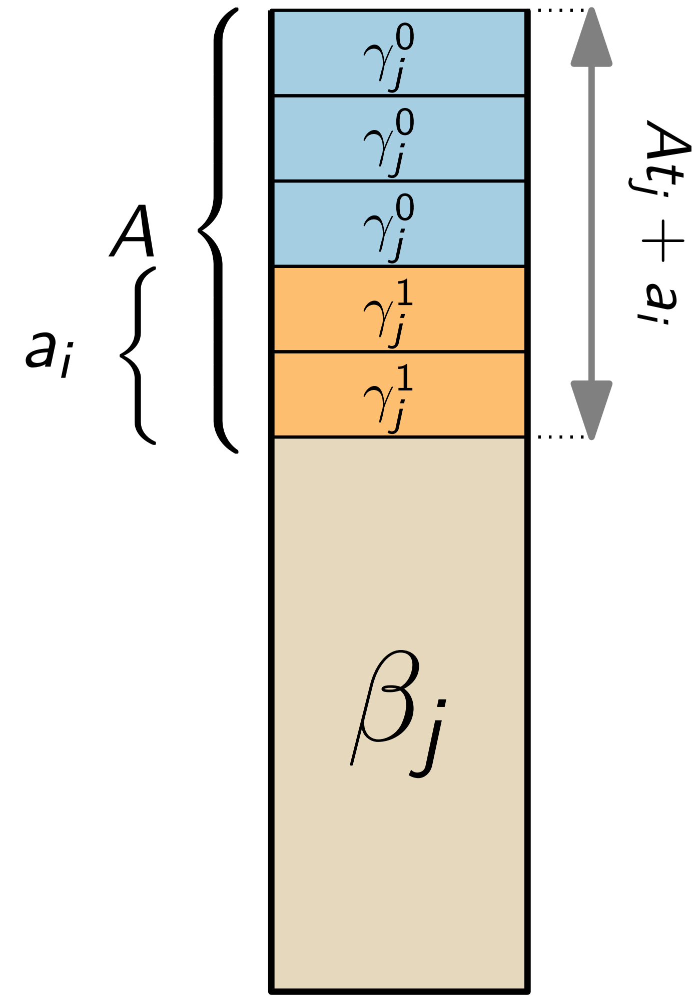
- $p_{\gamma_j^0} = t_j$
- $p_{\gamma_j^1} = t_j + 1$

$\Rightarrow$ $\leq a_i$ jobs of type γ_j^1 per M_i

$\geq A - a_i$ jobs of type γ_j^0 per M_i

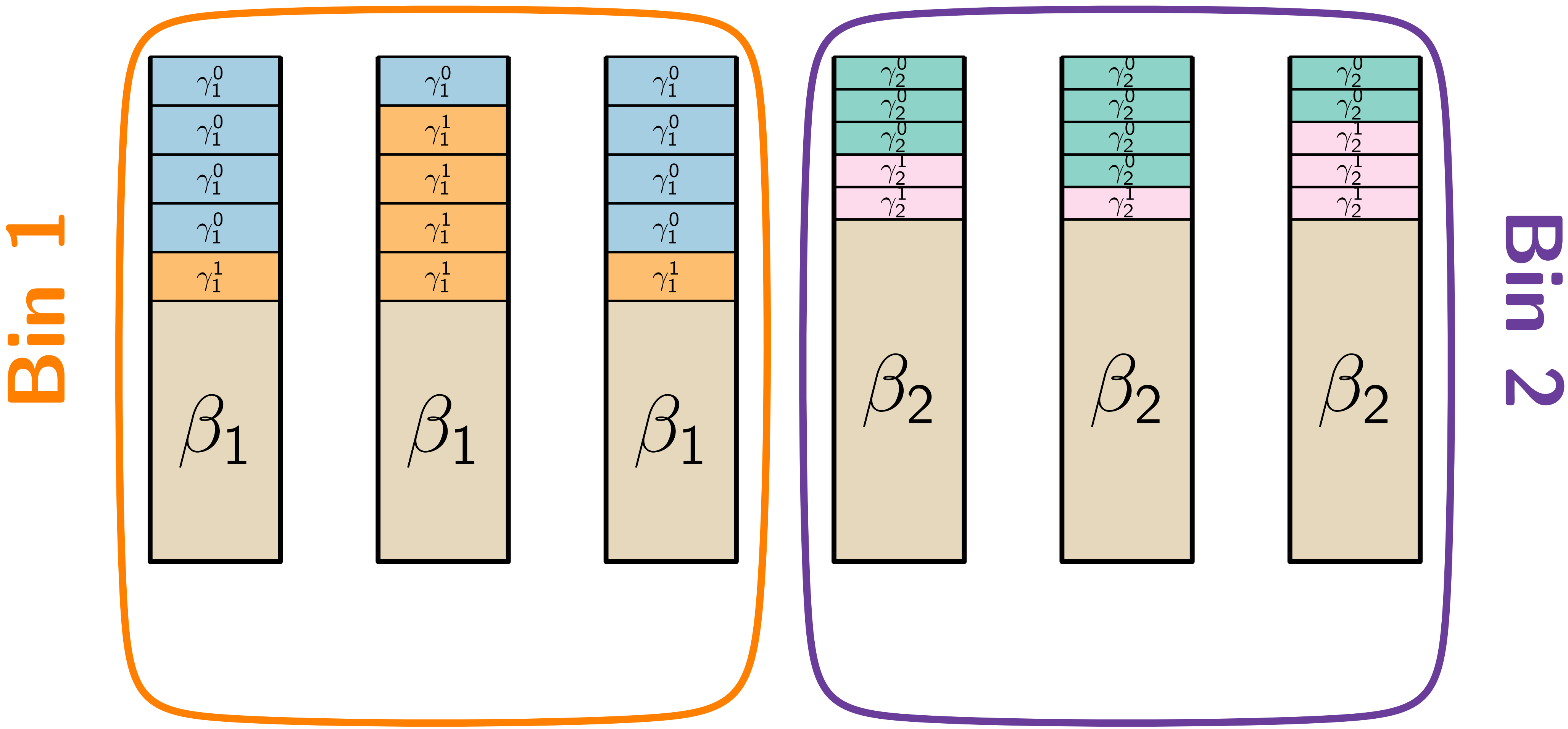
- We have $A = \sum_{i=0}^{m} a_i$ jobs of type $\gamma_\times^1$ in total.

$\Rightarrow$ M_i has exactly a_i jobs of type γ_j^1

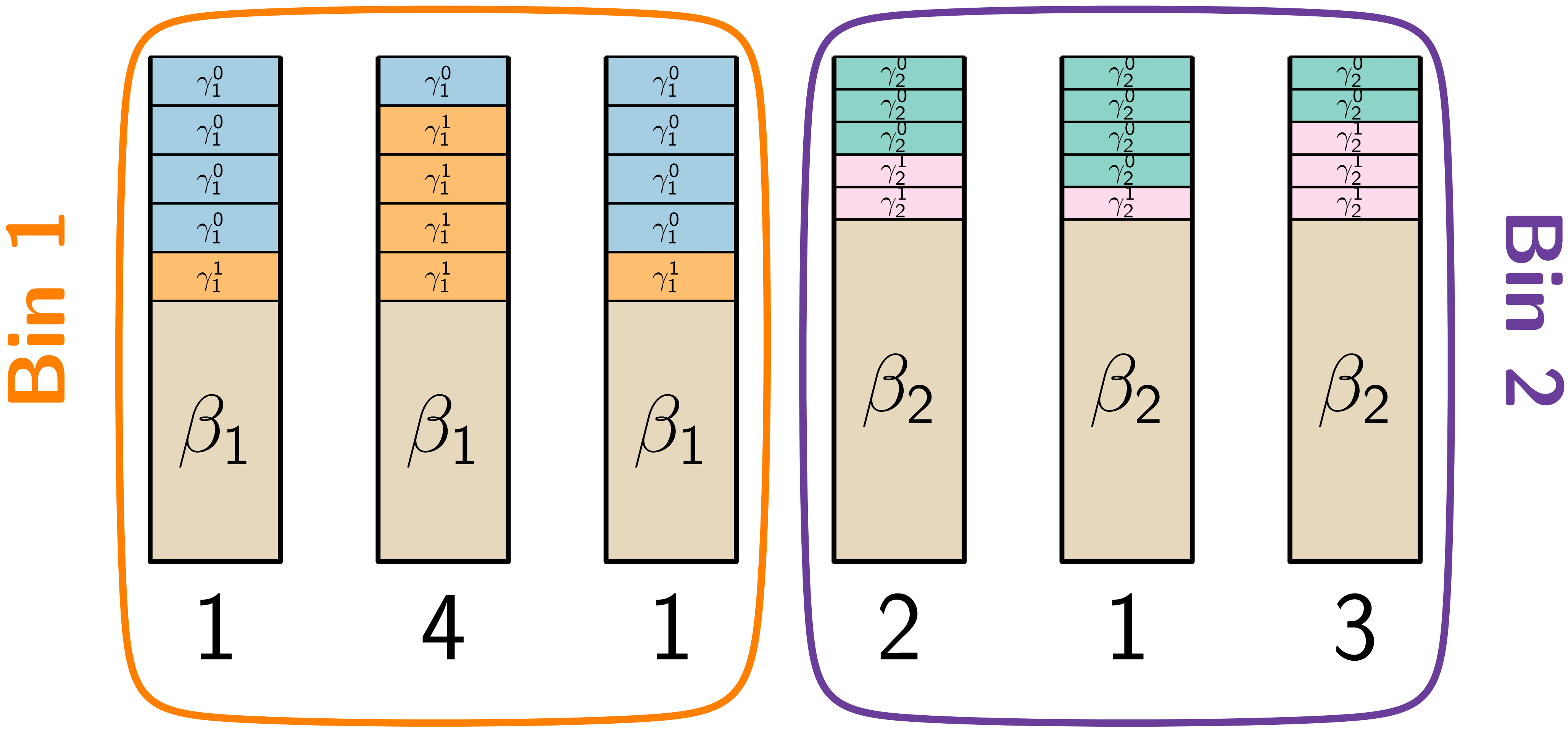


Machine M_i

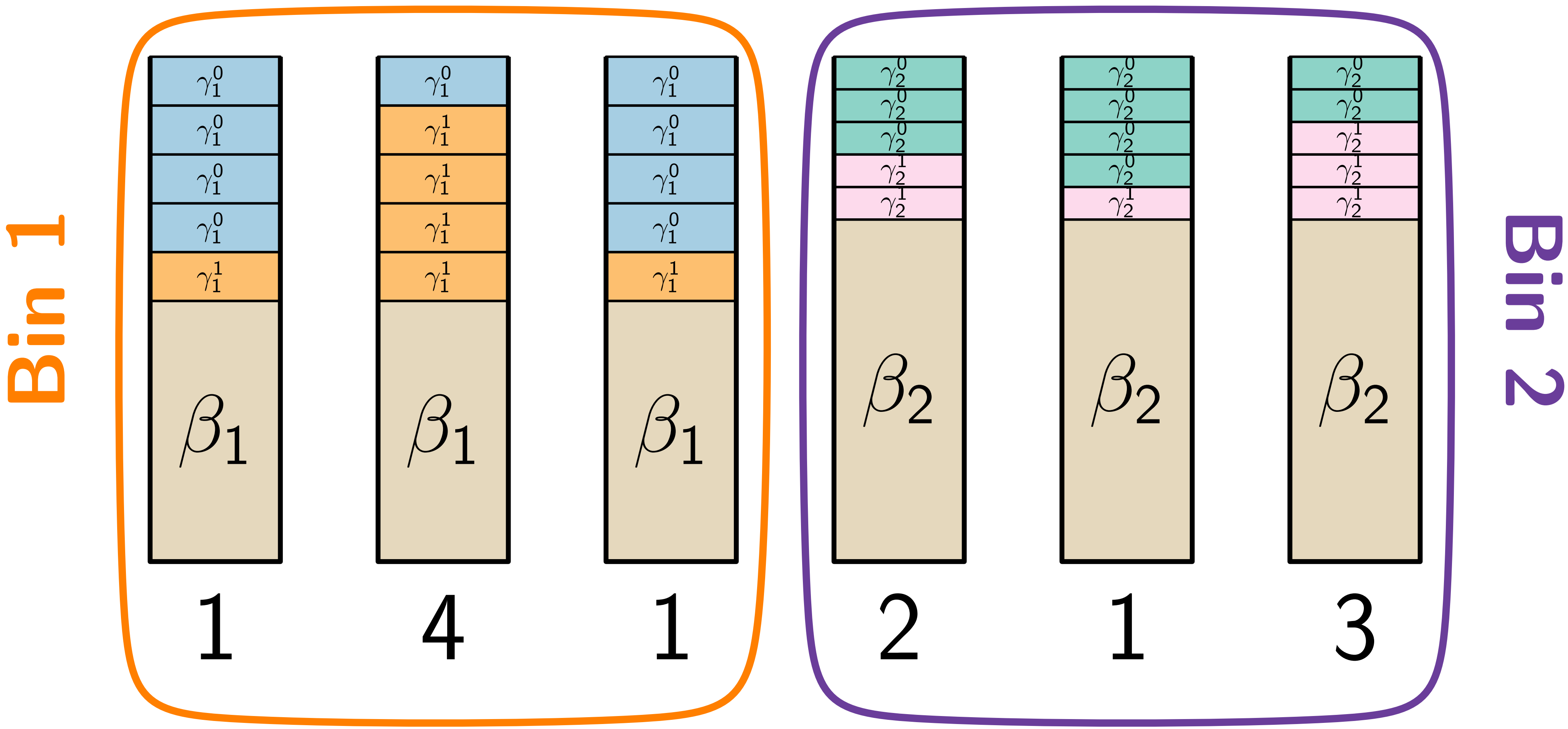
Balanced Bin Packing $\rightarrow Q|HM|C_{\max}$



Balanced Bin Packing $\to Q|HM|C_{\max}$



Balanced Bin Packing → $Q|HM|C_{\max}$



For $k = 2$ bins, Balanced Bin Packing is already NP-hard (→ Partition), there we use $2 \cdot 3 = 6$ job types.

Hardness

$Q|HM|C_{\max}$ is NP-hard already for 6 job types.

Hardness

$Q|HM|C_{\mathsf{max}}$ is NP-hard already for 6 job types.

And with some small modifications of this reduction:

Hardness

$Q|HM|C_{\max}$ is NP-hard already for 6 job types.

And with some small modifications of this reduction:

$R|HM|C_{\max}$ is NP-hard already for 4 job types.

Hardness

$Q|HM|C_{\max}$ is NP-hard already for 6 job types.

And with some small modifications of this reduction:

$R|HM|C_{\max}$ is NP-hard already for 4 job types.

Cutting Stock is NP-hard already with 8 item types.

Hardness

$Q|HM|C_{\max}$ is NP-hard already for 6 job types.

And with some small modifications of this reduction:

$R|HM|C_{\max}$ is NP-hard already for 4 job types.

CUTTING STOCK is NP-hard already with 8 item types.

$\{R, Q\}||C_{\max}$ is W[1]-hard parameterized by the number of job types even for unary input.

Hardness

$Q|HM|C_{\max}$ is NP-hard already for 6 job types.

And with some small modifications of this reduction:

$R|HM|C_{\max}$ is NP-hard already for 4 job types.

Cutting Stock is NP-hard already with 8 item types.

$\{R, Q\}||C_{\max}$ is W[1]-hard parameterized by the number of job types even for unary input.

Beside $C_{\max}$, most of our hardness results also hold true for the objectives ℓ_2 and $\sum w_j C_j$.

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \operatorname{poly}\log(n)$

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \operatorname{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \operatorname{poly}\log(n)$.

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \operatorname{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \operatorname{poly}\log(n)$.

FPT if the largest job size $p_{\max}$ is small

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \operatorname{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \operatorname{poly}\log(n)$.

FPT if the largest job size $p_{\max}$ is small

But our vector p of processing times for jobs may contain large numbers!

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \text{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \text{poly}\log(n)$.

FPT if the largest job size $p_{\max}$ is small

But our vector p of processing times for jobs may contain large numbers!

- Use coefficient reduction algorithm by Frank and Tardos. This yields a vector $\bar{\mathbf{p}}$, which is equivalent for comparisons, but its largest entry is in order n^k.

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \text{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \text{poly}\log(n)$.

  FPT if the largest job size $p_{\max}$ is small

  But our vector p of processing times for jobs may contain large numbers!

- Use coefficient reduction algorithm by Frank and Tardos. This yields a vector $\bar{\mathbf{p}}$, which is equivalent for comparisons, but its largest entry is in order n^k.

  $(\mathbf{p}, C_{\max})$

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \operatorname{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \operatorname{poly}\log(n)$.

FPT if the largest job size $p_{\max}$ is small

But our vector p of processing times for jobs may contain large numbers!

$\uparrow$

$(\bar{\mathbf{p}}, \bar{C}_{\max})$

$\uparrow$

- Use coefficient reduction algorithm by Frank and Tardos. This yields a vector $\bar{\mathbf{p}}$, which is equivalent for comparisons, but its largest entry is in order n^k.

$\uparrow$

$(\mathbf{p}, C_{\max})$

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \operatorname{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \operatorname{poly}\log(n)$.

$(\bar{\mathbf{p}}, \bar{C}_{\max})$

FPT if the largest job size $p_{\max}$ is small

But our vector p of processing times for jobs may contain large numbers!

- Use coefficient reduction algorithm by Frank and Tardos. This yields a vector $\bar{\mathbf{p}}$, which is equivalent for comparisons, but its largest entry is in order n^k.

m $(\mathbf{p}, C_{\max})$

Fixed-Parameter Tractability

$P||C_{\max}$ is FPT parameterized by the number of job types k.

Proof Idea:

assignment of jobs to machines

- Use algorithm by Goemans and Rothvoss, which has runtime $(\log p_{\max})^{f(k)} \cdot \text{poly}\log(n) \leq g(k) \cdot p_{\max}^{o(1)} \cdot \text{poly}\log(n)$.

$(\bar{\mathbf{p}}, \bar{C}_{\max})$

FPT if the largest job size $p_{\max}$ is small

But our vector p of processing times for jobs may contain large numbers!

- Use coefficient reduction algorithm by Frank and Tardos. This yields a vector $\bar{\mathbf{p}}$, which is equivalent for comparisons, but its largest entry is in order n^k.

m $(\mathbf{p}, C_{\max})$

Summary

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$
$C_{\max}$			

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

Summary

known before

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$
$C_{\max}$			

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

known before

Summary

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$
$C_{\max}$	poly. time for const. k		

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

Summary

known before
new

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$
$C_{\max}$	poly. time for const. k		

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

Summary

	known before
	new

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : $\#$ job types

Summary

known before
new

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$	$R\|\|\dots$
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$			

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

Summary

known before
new

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$	$P\|\|\ldots$	$Q\|\|\ldots$	$R\|\|\ldots$
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k		

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

Summary

known before
new

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$	$P\|\|\ldots$	$Q\|\|\ldots$	$R\|\|\ldots$
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	XP in k

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : $\#$ job types

Summary

known before
new

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard

P: identical machines

Q: related machines

R: unrelated machines

HM: high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2: min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k: # job types

Summary

known before
new
open

	$P \vert HM \vert \dots$	$Q \vert HM \vert \dots$	$R \vert HM \vert \dots$	$P \vert \vert \dots$	$Q \vert \vert \dots$		$R \vert \vert \dots$	
C_{max}	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard

P : identical machines

Q : related machines

R : unrelated machines

HM : high multipl. jobs

C_{max} : min. makespan

ℓ_2 : min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k : # job types

Summary

known before
new
open

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

P: identical machines

Q: related machines

R: unrelated machines

HM: high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2: min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k: # job types

Summary

known before
new
open

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

Open Problems

P: identical machines

Q: related machines

R: unrelated machines

HM: high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2: min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k: # job types

Summary

known before
new
open

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

Open Problems

- ? above in the table

P: identical machines

Q: related machines

R: unrelated machines

HM: high multipl. jobs

$C_{\max}$: min. makespan

ℓ_2: min. time under ℓ_2-Norm

$\sum w_j C_j$: min. sum of weighted completion times

k: # job types

Summary

known before
new
open

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

Open Problems

- ? above in the table
- Is it also NP-hard for fewer job types, e.g. $k = 2$ or 3?

Summary

known before
new
open

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$	$P\|\|\ldots$	$Q\|\|\ldots$		$R\|\|\ldots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

Open Problems

- ? above in the table
- Is it also NP-hard for fewer job types, e.g. $k = 2$ or 3?
- Is $P|HM|C_{\max}$ FPT in k or not?

Summary

known before
new
open

	$P\|HM\|\dots$	$Q\|HM\|\dots$	$R\|HM\|\dots$	$P\|\|\dots$	$Q\|\|\dots$		$R\|\|\dots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

Open Problems

- ? above in the table
- Is it also NP-hard for fewer job types, e.g. $k = 2$ or 3?
- Is $P|HM|C_{\max}$ FPT in k or not?
- Consider more constraints & obj. (release/due times, tardiness, ...)

Summary

known before
new
open

	$P\|HM\|\ldots$	$Q\|HM\|\ldots$	$R\|HM\|\ldots$	$P\|\|\ldots$	$Q\|\|\ldots$		$R\|\|\ldots$	
$C_{\max}$	poly. time for const. k	NP-hard for $k \geq 6$	NP-hard for $k \geq 4$	FPT in k	XP in k	W[1]-hard	XP in k	W[1]-hard
ℓ_2	?	NP-hard for $k \geq 6$	NP-hard for $k \geq 7$	?	XP	W[1]-hard	XP	W[1]-hard
$\sum w_j C_j$	?	?	NP-hard for $k \geq 7$	?	XP	?	XP	W[1]-hard

Open Problems

- ? above in the table
- Is it also NP-hard for fewer job types, e.g. $k = 2$ or 3?
- Is $P|HM|C_{\max}$ FPT in k or not?
- Consider more constraints & obj. (release/due times, tardiness, ...)
- Is Cutting Stock W[1]-hard when input is given in unary?