Optima Localization for Scheduling Problems: Computer-Aided Approach

Ilya Chernykh

Sobolev Institute of Mathematics, Novosibirsk

86th Workshop on Algorithms, Complexity and Logic
19.04.2024

The research was supported by
Russian Science Foundation grant N 22-71-10015

The lower bound and instances' properties

Let

$$F(x) \to \min$$

be some minimization problem, $LB \leqslant F^*$ is the lower bound on the optimuim.

Interesting properties of instances in terms of LB:

- Polynomially solvable subcases, for which $F^* = LB$;
- Approxination algorithms with ratio based on LB:

$$F(x_A) \leqslant \rho LB \leqslant \rho F^*;$$

- Optima localization.

Optima localization

Definition

Tight optima localization interval for a class of instances $\mathcal{I}$ of some minimization problem respective to some lower bound LB is the tightest possible interval of form

$$\mathtt{OL}_{LB}(\mathcal{I}) = [LB, \rho^* LB],$$

guaranteed to contain optima of all instances from $\mathcal{I}$.

Optima localization

Definition

Tight optima localization interval for a class of instances $\mathcal{I}$ of some minimization problem respective to some lower bound LB is the tightest possible interval of form

$$\mathtt{OL}_{LB}(\mathcal{I}) = [LB, \rho^* LB],$$

guaranteed to contain optima of all instances from $\mathcal{I}$.

$$\rho^* = \sup_{I \in \mathcal{I}} \alpha(I) = \sup_{I \in \mathcal{I}} \frac{\mathrm{OPT}(I)}{LB(I)}.$$

Optima localization

Definition

Tight optima localization interval for a class of instances $\mathcal{I}$ of some minimization problem respective to some lower bound LB is the tightest possible interval of form

$$\texttt{OL}_{LB}(\mathcal{I}) = [LB, \rho^* LB],$$

guaranteed to contain optima of all instances from $\mathcal{I}$.

$$\rho^* = \sup_{I \in \mathcal{I}} \alpha(I) = \sup_{I \in \mathcal{I}} \frac{\mathrm{OPT}(I)}{LB(I)}.$$

Motivation

1. Quality of lower bound LB.
2. Upper bound on the approximation ratio for algorithms, based on LB.
3. Potential for designing approximation algorithms with "best possible" approximation ratio (with respect to LB).

Shop scheduling problems

"General" problem settings

- A set of jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$,
- a set of machines $\mathcal{M} = \{M_1, \ldots, M_m\}$,
- each machine M_i performs single operation O_{ji} for each job J_j, processing times are given in advance

$$P = \begin{pmatrix} p_{11} & p_{21} & p_{31} & \ldots & p_{n1} \\ p_{12} & p_{22} & p_{32} & \ldots & p_{n2} \\ \vdots & \vdots & \vdots & & \vdots \\ p_{1m} & p_{2m} & p_{3m} & \ldots & p_{nm} \end{pmatrix},$$

- operations of the same job/machine cannot be performed simultaneously.

Shop scheduling problems

"General" problem settings

- A set of jobs $\mathcal{J} = \{J_1, \dots, J_n\}$,
- a set of machines $\mathcal{M} = \{M_1, \dots, M_m\}$,
- each machine M_i performs single operation O_{ji} for each job J_j, processing times are given in advance

$$P = \begin{pmatrix} p_{11} & p_{21} & p_{31} & \dots & p_{n1} \\ p_{12} & p_{22} & p_{32} & \dots & p_{n2} \\ \vdots & \vdots & \vdots & & \vdots \\ p_{1m} & p_{2m} & p_{3m} & \dots & p_{nm} \end{pmatrix},$$

- operations of the same job/machine cannot be performed simultaneously.

Standard lower bound on the makespan $C_{\max}$:

$$SLB = \max_{i,j}\{\ell_i, d_j\} = \max_{i,j}\left\{\sum_{j=1}^{n} p_{ji}, \sum_{i=1}^{m} p_{ji}\right\}.$$

Open shop problem

Notation

Open shop with m machines is denoted as $Om||C_{\max}$. We denote the set of instances as $\mathcal{I}_m$.

Open shop problem

Notation

Open shop with m machines is denoted as $Om||C_{\max}$. We denote the set of instances as $\mathcal{I}_m$.

Some known facts

1. $\mathtt{OL}_{SLB}(\mathcal{I}_2) = [SLB, SLB]$ [Gonzalez, Sahni 1976],
2. $\mathtt{OL}_{SLB}(\mathcal{I}_3) = [SLB, \frac{4}{3}SLB]$ [Sevastyanov, Ch 1998].

Open shop problem

Notation

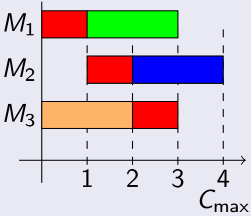
Open shop with m machines is denoted as $Om||C_{\max}$. We denote the set of instances as $\mathcal{I}_m$.

Some known facts

1. $\mathtt{OL}_{SLB}(\mathcal{I}_2) = [SLB, SLB]$ [Gonzalez, Sahni 1976],
2. $\mathtt{OL}_{SLB}(\mathcal{I}_3) = [SLB, \frac{4}{3}SLB]$ [Sevastyanov, Ch 1998].

Critical instance for $m = 3$

$$\begin{pmatrix} 1 & 2 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 1 & 0 & 0 & 2 \end{pmatrix}, SLB = 3.$$



Early schedules

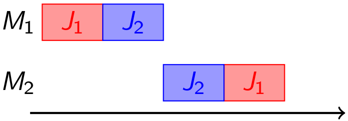
Definition

Schedule S is early, if no operation can be started earlier than in S, without the violation of feasibility and precedence induced by S.

Early schedules

Definition

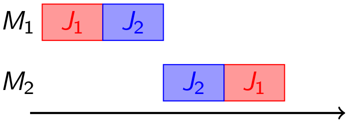
Schedule S is early, if no operation can be started earlier than in S, without the violation of feasibility and precedence induced by S.

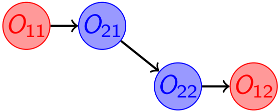


Early schedules

Definition

Schedule S is early, if no operation can be started earlier than in S, without the violation of feasibility and precedence induced by S.

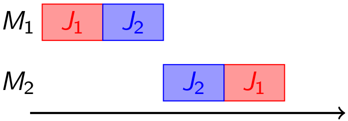


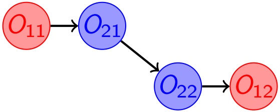


Early schedules

Definition

Schedule S is early, if no operation can be started earlier than in S, without the violation of feasibility and precedence induced by S.





The set of all possible early schedules contains an optimal one. An early schedule can be described with linear ordering of operations for each job and each machine.

Schedule's templates

Definition

A schedule's template for a problem with m machines and n job is a partial order of operations, such that operations of each job and each machine are linearly ordered.

Schedule's templates

Definition

A schedule's template for a problem with m machines and n job is a partial order of operations, such that operations of each job and each machine are linearly ordered.

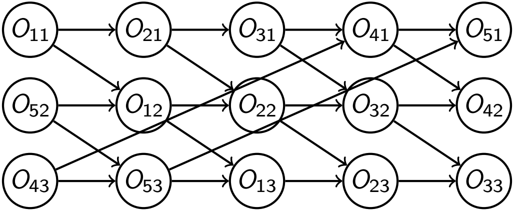
A combination of a template and an instance defines a single early schedule. Its makespan equals to the weight of a critical path.

Schedule's templates

Definition

A schedule's template for a problem with m machines and n job is a partial order of operations, such that operations of each job and each machine are linearly ordered.

A combination of a template and an instance defines a single early schedule. Its makespan equals to the weight of a critical path.

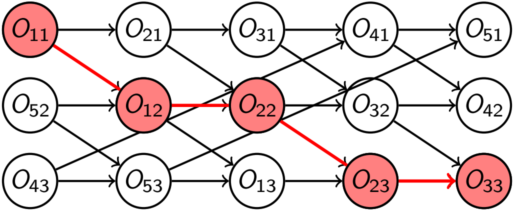


Schedule's templates

Definition

A schedule's template for a problem with m machines and n job is a partial order of operations, such that operations of each job and each machine are linearly ordered.

A combination of a template and an instance defines a single early schedule. Its makespan equals to the weight of a critical path.



Job aggregation

Definition

Job aggregation of a subset of jobs K of instance I is the following transformation $I \to I'$:

$$\mathcal{J}(I') = \mathcal{J}(I) \setminus K \cup \{J_K\},$$

$$\forall i = 1, \ldots, m \quad p_K i = \sum_{J_j \in K} p_{ji}.$$

Job aggregation

Definition

Job aggregation of a subset of jobs K of instance I is the following transformation $I \to I'$:

$$\mathcal{J}(I') = \mathcal{J}(I) \setminus K \cup \{J_K\},$$

$$\forall i = 1, \ldots, m \quad p_K i = \sum_{J_j \in K} p_{ji}.$$

$$P = \begin{pmatrix} p_{11} & p_{21} & \cdots & p_{j-1,1} & p_{j1} & \cdots & p_{r1} & p_{r+1,1} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{j-1,2} & p_{j2} & \cdots & p_{r2} & p_{r+1,2} & \cdots & p_{n2} \\ p_{13} & p_{23} & \cdots & p_{j-1,3} & p_{j3} & \cdots & p_{r3} & p_{r+1,3} & \cdots & p_{n3} \\ \vdots & \vdots & & & & & & & & \vdots \\ p_{1m} & p_{2m} & \cdots & p_{j-1,m} & p_{jm} & \cdots & p_{rm} & p_{r+1,m} & \cdots & p_{nm} \end{pmatrix}$$

Job aggregation

Definition

Job aggregation of a subset of jobs K of instance I is the following transformation $I \to I'$:

$$\mathcal{J}(I') = \mathcal{J}(I) \setminus K \cup \{J_K\},$$

$$\forall i = 1, \dots, m \quad p_K i = \sum_{J_j \in K} p_{ji}.$$

$$P = \begin{pmatrix} p_{11} & p_{21} & \cdots & p_{j-1,1} & p_{j1} & \cdots & p_{r1} & p_{r+1,1} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{j-1,2} & p_{j2} & \cdots & p_{r2} & p_{r+1,2} & \cdots & p_{n2} \\ p_{13} & p_{23} & \cdots & p_{j-1,3} & p_{j3} & \cdots & p_{r3} & p_{r+1,3} & \cdots & p_{n3} \\ \vdots & \vdots & & & & & & & & \vdots \\ p_{1m} & p_{2m} & \cdots & p_{j-1,m} & p_{jm} & \cdots & p_{rm} & p_{r+1,m} & \cdots & p_{nm} \end{pmatrix}$$

Job aggregation

Definition

Job aggregation of a subset of jobs K of instance I is the following transformation $I \to I'$:

$$\mathcal{J}(I') = \mathcal{J}(I) \setminus K \cup \{J_K\},$$

$$\forall i = 1, \ldots, m \quad p_K i = \sum_{J_j \in K} p_{ji}.$$

$$P = \begin{pmatrix} p_{11} & p_{21} & \cdots & p_{j-1,1} & \cdots & p_{r+1,1} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{j-1,2} & \cdots & p_{r+1,2} & \cdots & p_{n2} \\ p_{13} & p_{23} & \cdots & p_{j-1,3} & \cdots & p_{r+1,3} & \cdots & p_{n3} \\ \vdots & \vdots & & & & & & \vdots \\ p_{1m} & p_{2m} & \cdots & p_{j-1,m} & \cdots & p_{r+1,m} & \cdots & p_{nm} \end{pmatrix}$$

Job aggregation

Definition

Job aggregation of a subset of jobs K of instance I is the following transformation $I \to I'$:

$$\mathcal{J}(I') = \mathcal{J}(I) \setminus K \cup \{J_K\},$$

$$\forall i = 1, \ldots, m \quad p_K i = \sum_{J_j \in K} p_{ji}.$$

$$P = \begin{pmatrix} p_{11} & p_{21} & \cdots & p_{j-1,1} & \sum_{k=j}^{r} p_{k1} & p_{r+1,1} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{j-1,2} & \sum_{k=j}^{r} p_{k2} & p_{r+1,2} & \cdots & p_{n2} \\ p_{13} & p_{23} & \cdots & p_{j-1,3} & \sum_{k=j}^{r} p_{k3} & p_{r+1,3} & \cdots & p_{n3} \\ \vdots & \vdots & & & & & & \vdots \\ p_{1m} & p_{2m} & \cdots & p_{j-1,m} & \sum_{k=j}^{r} p_{km} & p_{r+1,m} & \cdots & p_{nm} \end{pmatrix}$$

Job aggregation

Definition

Job aggregation of a subset of jobs K of instance I is the following transformation $I \to I'$:

$$\mathcal{J}(I') = \mathcal{J}(I) \setminus K \cup \{J_K\},$$

$$\forall i = 1, \ldots, m \quad p_K i = \sum_{J_j \in K} p_{ji}.$$

$$P = \begin{pmatrix} p_{11} & p_{21} & \cdots & p_{j-1,1} & \sum_{k=j}^{r} p_{k1} & p_{r+1,1} & \cdots & p_{n1} \\ p_{12} & p_{22} & \cdots & p_{j-1,2} & \sum_{k=j}^{r} p_{k2} & p_{r+1,2} & \cdots & p_{n2} \\ p_{13} & p_{23} & \cdots & p_{j-1,3} & \sum_{k=j}^{r} p_{k3} & p_{r+1,3} & \cdots & p_{n3} \\ \vdots & \vdots & & & & & & \vdots \\ p_{1m} & p_{2m} & \cdots & p_{j-1,m} & \sum_{k=j}^{r} p_{km} & p_{r+1,m} & \cdots & p_{nm} \end{pmatrix}$$

$$SLB(I') = SLB(I) \iff \sum_{J_j \in K} d_j \leqslant SLB(I).$$

Aggregation theorem

Theorem

Any instance I of an m-machine problem can be transformed by a series of aggregations into instance I' such that $SLB(I') = SLB(I)$ and I' contains at most $2m - 1$ jobs.

Aggregation theorem

Theorem

Any instance I of an m-machine problem can be transformed by a series of aggregations into instance I' such that $SLB(I') = SLB(I)$ and I' contains at most $2m - 1$ jobs.

Proof

It is sufficient to show that we can always group the vaslues $d_1, \ldots, d_n$ in at most $2m - 1$ groups such that total value of each group doesn't exceed SLB.

$$\sum_{j=1}^{n} d_j = \sum_{i=1}^{m} \ell_i \leqslant mSLB.$$

While $n \geqslant 2m$: consider m pairs of values d_j. The sum of at least one of those pairs is $\leqslant SLB$.

Example of the usage: classical result for $O2||C_{\max}$

Gonzalez-Sahni theorem (1976)

For the problem $O2||C_{\max}$ the optimal makespan always coincides with SLB. Such (optimal) schedule can be built in $O(n)$ time.

Example of the usage: classical result for $O2||C_{max}$

Gonzalez-Sahni theorem (1976)

For the problem $O2||C_{max}$ the optimal makespan always coincides with SLB. Such (optimal) schedule can be built in $O(n)$ time.

- Apply job aggregations to I to obtain an instance I' with at most three jobs, preserving SLB.

Example of the usage: classical result for $O2||C_{\max}$

Gonzalez-Sahni theorem (1976)

For the problem $O2||C_{\max}$ the optimal makespan always coincides with SLB. Such (optimal) schedule can be built in $O(n)$ time.

- Apply job aggregations to I to obtain an instance I' with at most three jobs, preserving SLB.
- Denote operations of the first (second) machine by a_j (b_j), $j = 1, 2, 3$.

Example of the usage: classical result for $O2||C_{\max}$

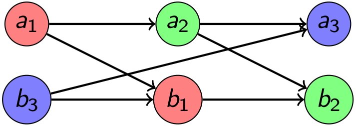
Gonzalez-Sahni theorem (1976)

For the problem $O2||C_{\max}$ the optimal makespan always coincides with SLB. Such (optimal) schedule can be built in $O(n)$ time.

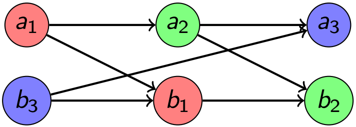
- Apply job aggregations to I to obtain an instance I' with at most three jobs, preserving SLB.
- Denote operations of the first (second) machine by a_j (b_j), $j = 1, 2, 3$.
- Consider two cases:
  - either $\forall j = 1, 2, 3\; a_j \geqslant b_j$ (equivalently $\forall j = 1, 2, 3\; a_j \leqslant b_j$)
  - or we have one false and two true inequalities (equivalently one true and two false).

- **Case I:** $\forall j = 1, 2, 3 \; a_j \geqslant b_j$.
  Without loss of generality $a_3 = \max\{a_j\}$.

- **Case I:** $\forall j = 1, 2, 3\ a_j \geqslant b_j$.
  Without loss of generality $a_3 = \max\{a_j\}$.
- Construct the schedule with the following template:



- **Case I:** $\forall j = 1, 2, 3\ a_j \geqslant b_j$.
  Without loss of generality $a_3 = \max\{a_j\}$.
- Construct the schedule with the following template:

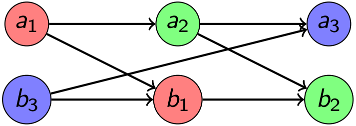


- Because of $a_3 \geqslant a_2 \geqslant b_2$ and $a_2 + a_3 \geqslant a_2 + a_1 \geqslant b_1 + b_2$, the makespan doesn't exceed SLB.

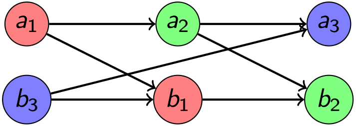
- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.

- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.
- **Case IIa:** $b_3 \geqslant a_1$.

- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.
- **Case IIa:** $b_3 \geqslant a_1$.
- Use the following template:



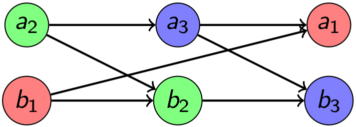
- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.
- **Case IIa:** $b_3 \geqslant a_1$.
- Use the following template:



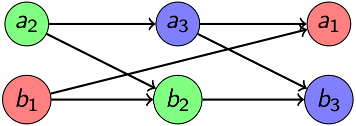
- As soon as $a_1 \leqslant b_3$ and $a_3 \geqslant b_3 \geqslant b_2$, the makespan doesn't exceed SLB.

- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.
- **Case IIb:** $b_3 \leqslant a_1$.

- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.
- **Case IIb:** $b_3 \leqslant a_1$.
- Use the following template:



- **Case II:** $a_1 \leqslant b_1$, $a_2 \geqslant b_2$, $a_3 \geqslant b_3$.
  Without loss of generality $b_3 \geqslant b_2$.
- **Case IIb:** $b_3 \leqslant a_1$.
- Use the following template:



- Because of $a_1 \geqslant b_3$ and $a_3 \geqslant b_3 \geqslant b_2$, the makespan doesn't exceed SLB.

de Werra's algorithm (1989)

Algorithm

de Werra's algorithm (1989)

Algorithm

1. If $\exists k | a_k + b_k = \bar{C}$, then the following schedule is optimal:

Algorithm

1. If $\exists k | a_k + b_k = \bar{C}$, then the following schedule is optimal:

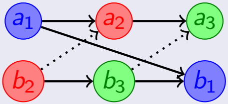
2. Otherwise: without loss of generality
   - Assume $a_1 + a_2 + a_3 = b_1 + b_2 + b_3$,
   - if necessary, rename jobs/machines such that $a_1 \geqslant b_2$ and $b_1 \geqslant a_3$.

de Werra's algorithm (1989)

Algorithm

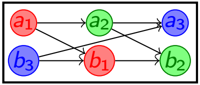
1. If $\exists k | a_k + b_k = \bar{C}$, then the following schedule is optimal:

2. Otherwise: without loss of generality
   - Assume $a_1 + a_2 + a_3 = b_1 + b_2 + b_3$,
   - if necessary, rename jobs/machines such that $a_1 \geqslant b_2$ and $b_1 \geqslant a_3$.
3. Construct the schedule:



Review of the "inefficient" version

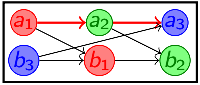
$$a_3 = \max_j \{a_j, b_j\}$$



S_1

Review of the "inefficient" version

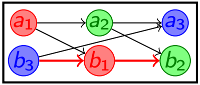
$$a_3 = \max_j \{a_j, b_j\}$$



S_1

Review of the “inefficient” version

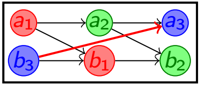
$a_3 = \max_j \{a_j, b_j\}$



S_1

Review of the "inefficient" version

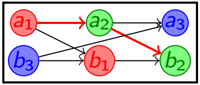
$$a_3 = \max_j \{a_j, b_j\}$$



S_1

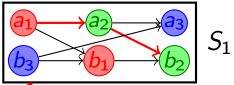
Review of the “inefficient” version

$$a_3 = \max_j \{a_j, b_j\}$$



S_1

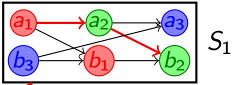
$$a_3 = \max_j \{a_j, b_j\}$$



$$C_{11} = a_1 + a_2 + b_2$$

Review of the “inefficient” version

$a_3 = \max_j\{a_j, b_j\}$

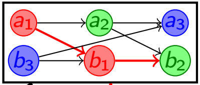


$C_{11} = a_1 + a_2 + b_2$

$C_{11} \leqslant \ell_1 \leqslant SLB$

Review of the "inefficient" version

$$a_3 = \max_j \{a_j, b_j\}$$



S_1

$$C_{11} = a_1 + a_2 + b_2$$

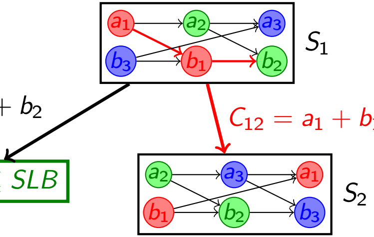
$$C_{12} = a_1 + b_1 + b_2$$

$$\boxed{C_{11} \leqslant \ell_1 \leqslant SLB}$$

$$a_3 = \max_j\{a_j, b_j\}$$

$$C_{11} = a_1 + a_2 + b_2$$

$$\boxed{C_{11} \leqslant \ell_1 \leqslant SLB}$$



$$C_{12} = a_1 + b_1 + b_2$$

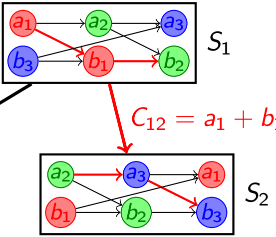
Review of the "inefficient" version

$$a_3 = \max_j \{a_j, b_j\}$$

$$C_{11} = a_1 + a_2 + b_2$$

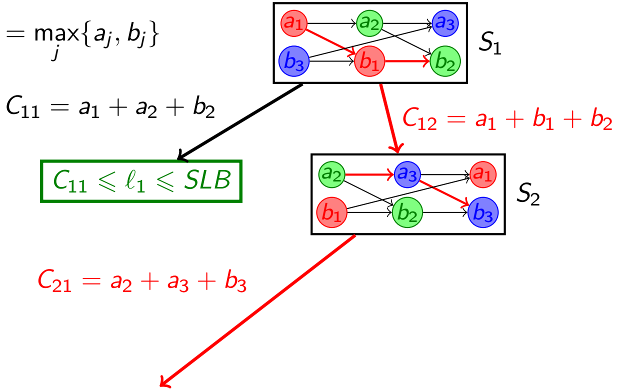
$$C_{11} \leqslant \ell_1 \leqslant SLB$$

$$C_{12} = a_1 + b_1 + b_2$$



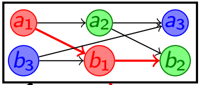
Review of the "inefficient" version

$a_3 = \max_j \{a_j, b_j\}$



Review of the "inefficient" version

$$a_3 = \max_j \{a_j, b_j\}$$

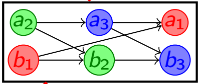


S_1

$$C_{11} = a_1 + a_2 + b_2$$

$$C_{12} = a_1 + b_1 + b_2$$

$$C_{11} \leqslant \ell_1 \leqslant SLB$$



S_2

$$C_{21} = a_2 + a_3 + b_3$$

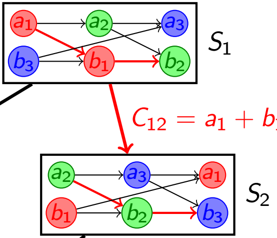
$$C_{12} + C_{21} = \ell_1 + \ell_2 \leqslant 2SLB$$

Review of the "inefficient" version

$$a_3 = \max_j\{a_j, b_j\}$$

$$C_{11} = a_1 + a_2 + b_2$$

$$C_{11} \leqslant \ell_1 \leqslant SLB$$

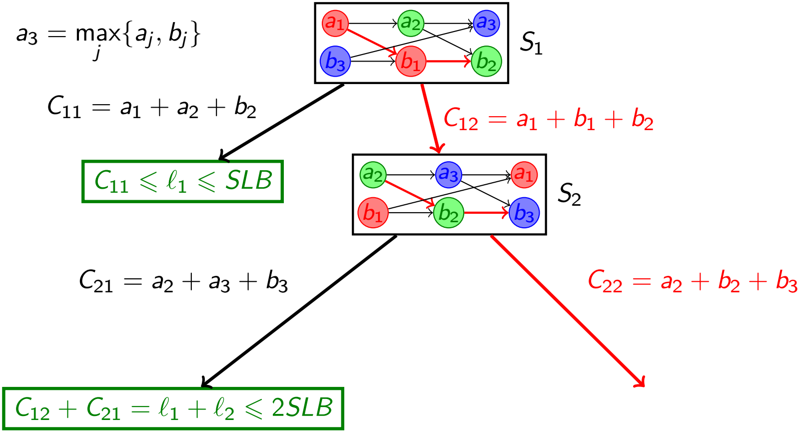


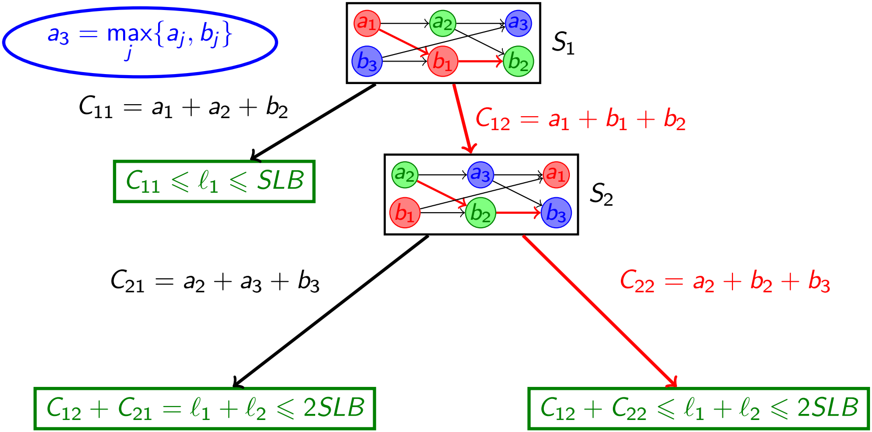
$$C_{12} = a_1 + b_1 + b_2$$

$$C_{21} = a_2 + a_3 + b_3$$

$$C_{12} + C_{21} = \ell_1 + \ell_2 \leqslant 2SLB$$

Review of the “inefficient” version

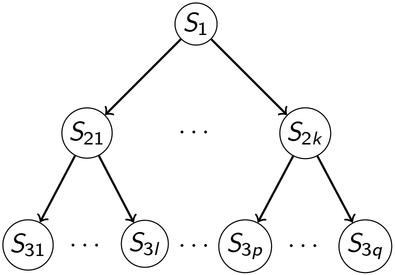




$a_3 = \max_j \{a_j, b_j\}$
a_1 a_2 a_3 b_3 b_1 b_2
S_1
$C_{11} = a_1 + a_2 + b_2$
$C_{12} = a_1 + b_1 + b_2$
$C_{11} \leqslant \ell_1 \leqslant SLB$
a_2 a_3 a_1 b_1 b_2 b_3
S_2
$C_{21} = a_2 + a_3 + b_3$
$C_{22} = a_2 + b_2 + b_3$
$C_{12} + C_{21} = \ell_1 + \ell_2 \leqslant 2SLB$
$C_{12} + C_{22} \leqslant \ell_1 + \ell_2 \leqslant 2SLB$

A tree of proof

- Vertices (except sinks) are *templates*.
- Arcs correspond to variants of critical paths in the templates they leave.
- Sinks (terminal vertices) contain *proofs*, that at least one of the schedules, built according to the templates belonging to the path from the root to this sink is good enough.



Automation

1. It is sufficient to consider instances with at most $2m - 1$ jobs.
2. It is sufficient to consider instances with $SLB = 1$.
3. Questions:
   - How to choose the next template when branching?
   - How to prove that no further branching is necessary?

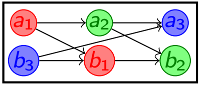
Automation

1. It is sufficient to consider instances with at most $2m - 1$ jobs.
2. It is sufficient to consider instances with $SLB = 1$.
3. Questions:
   - How to choose the next template when branching?
   - How to prove that no further branching is necessary?

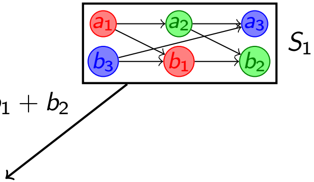
General ideas

1. Treat the branching as splitting of the set of instances. Now each node corresponds to a subset of instances.
2. Find a critical instance in each subset (with maximal makespan for the subset).
3. If the makespan if critical instance is good enough, we have the proof (for this vertex).
4. Otherwise, choose the template that suits the critical instance the most.

Example ($SLB = 1$)

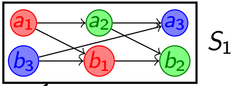


S_1



$C_1 = a_1 + b_1 + b_2$

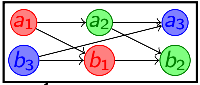
Example ($SLB = 1$)



$C_1 = a_1 + b_1 + b_2$

$a_1 = b_2 = 1$

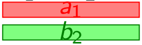
Example ($SLB = 1$)



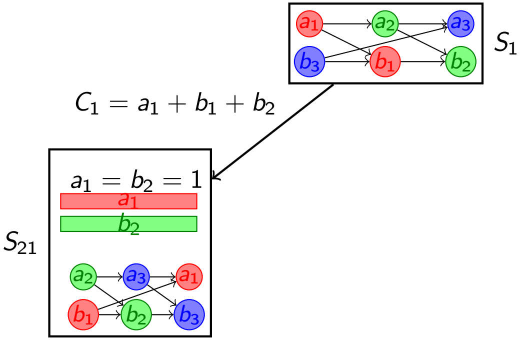
S_1

$C_1 = a_1 + b_1 + b_2$

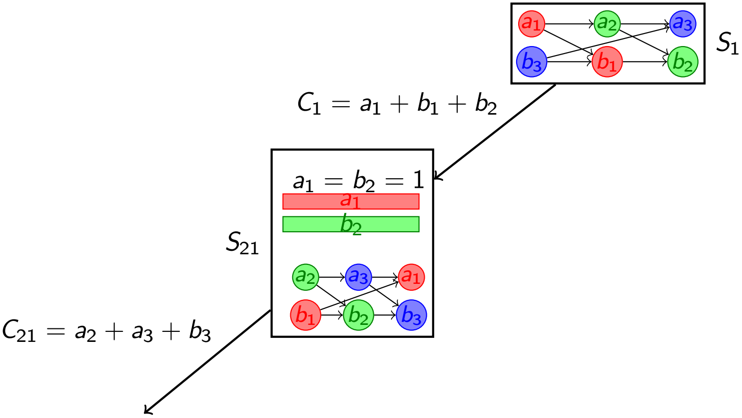
$a_1 = b_2 = 1$



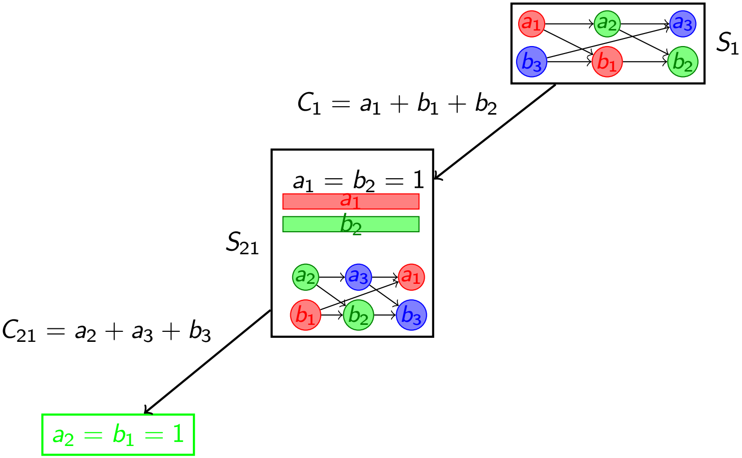
Example ($SLB = 1$)



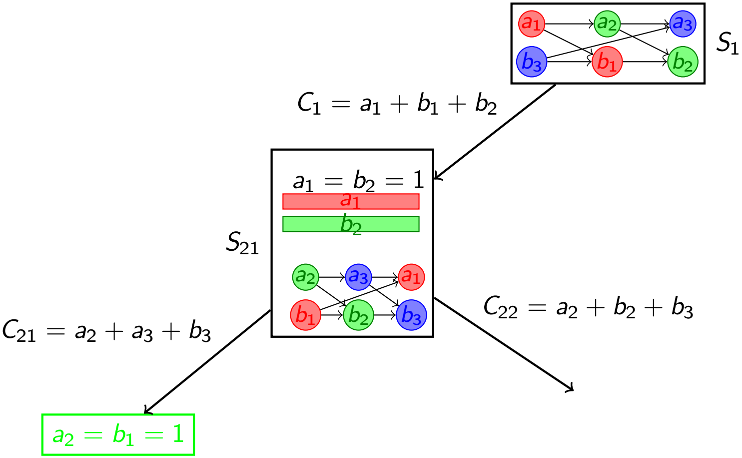
Example ($SLB = 1$)



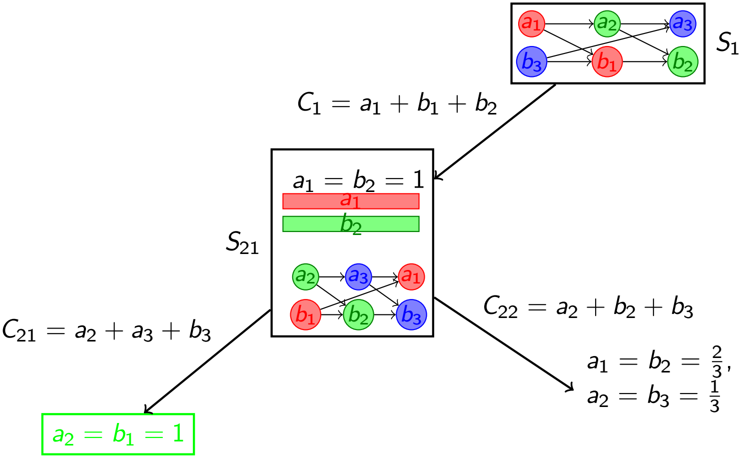
Example ($SLB = 1$)



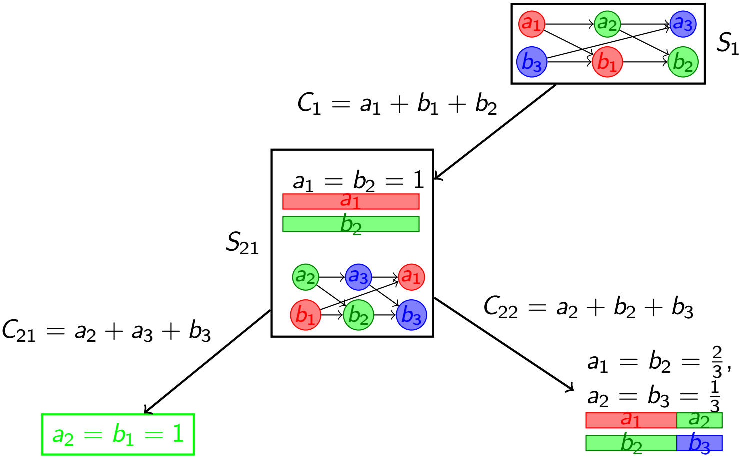
Example ($SLB = 1$)



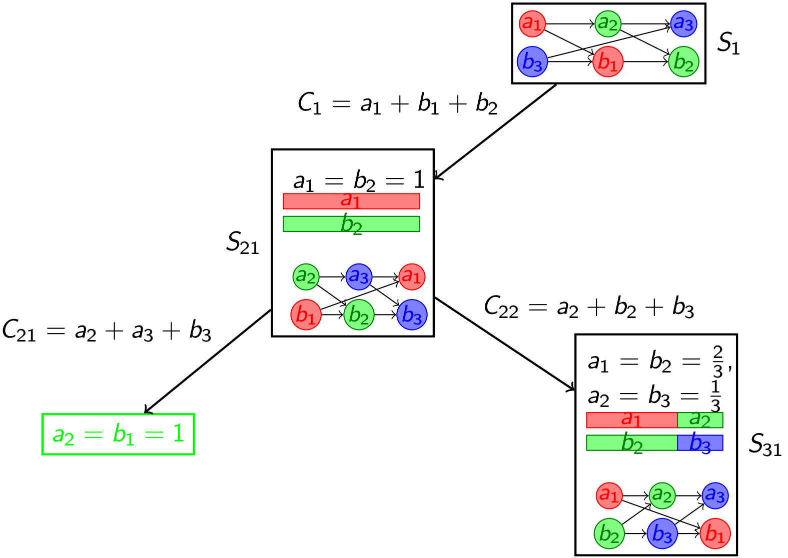
Example ($SLB = 1$)



Example ($SLB = 1$)



Example ($SLB = 1$)



How to find a critical instance

Linear programming

- Nonnegative variables: processing times (a_j, b_j) and auxiliary variable ρ.
- Objective: $\rho \to \max$.
- Subject to:

$$\begin{cases} \ell_1 = a_1 + a_2 + a_3 \leqslant 1, \\ \ell_2 = b_1 + b_2 + b_3 \leqslant 1, \\ d_1 = a_1 + b_1 \leqslant 1, \\ d_2 = a_2 + b_2 \leqslant 1, \\ d_3 = a_3 + b_3 \leqslant 1, \\ C_1 = \sum_{O \in P_1} p(O) \geqslant \rho, \\ \dots \\ C_k = \sum_{O \in P_k} p(O) \geqslant \rho, \\ a_j, b_j, \rho \geqslant 0. \end{cases}$$

Further automation

How to choose the next templete

Possible variants:

1. Critical instance $\to$ optimal schedule $\to$ partial order of the operations $\to$ template.
2. Create some pool of instances. Choose the best fitting the critical instance from the pool.

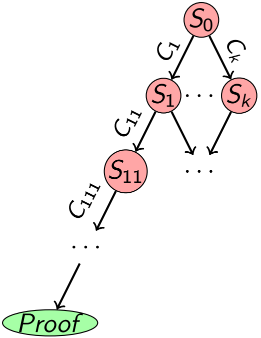
Optimization

- "Base" templates can be modified with application of permutations of jobs/machines.
- Introduce properties of an instance without loss of generality (*e.g.*, $a_3 = \max\{a_j, b_j\}$).
- Split $\mathcal{I}$ into subcases (by number of jobs, by total workload, ...)

Verifying the proof constructed

- The proof is constructive.
- We need a verification for each terminal vertex.
- We could verify by solving the corresponding LP (for terminal vertices only).
- Faster approach: for each terminal vertex store an optimal solution of dual LP!

The structure of the proof



- Vertices correspond to subsets of instances
- red vertices contain templates
- arcs, outcoming from red vertices, correspond to variants of critical paths in that templates. Each path is stored as a linear expression
- terminal (green) vertices contain proofs

Algorithm: building the tree of proof

Conjecture: For any instance (with certain restrictions) there exists schedule S with makespan $C_{\max}(S) \leqslant \rho^* SLB$.

- Given a set of templates $\mathcal{S} = \{S_0, \ldots, S_N\}$.
- For each template S_k the set $\mathcal{P}(S_k)$ of possible critical paths is described.

1. Root vertex: $S := S_0$. Current set of paths: $Q = \emptyset$.
2. For each $P \in \mathcal{P}(S)$:
    1. $Q := Q \cup \{P\}$.
    2. Find a critical instance $\tilde{I} = \arg\max_I \min_{P \in Q} P(I)$.
    3. If $\min_{P \in Q} P(\tilde{I}) \leqslant \rho^*$, $Q := Q \setminus \{P\}$, continue to the next P, if any.
       Otherwise climb up to the previous vertex, continue to the next P.
    4. Else
        1. Find an improving template $S \in \mathcal{S}$ for the instance $\tilde{I}$, Go to Step 2.
        2. If improving template is not found, STOP, output the critical instance.
3. Done!

Algorithm: building the tree of proof

Conjecture: For any instance (with certain restrictions) there exists schedule S with makespan $C_{\max}(S) \leqslant \rho^* SLB$.

- Given a set of templates $\mathcal{S} = \{S_0, \ldots, S_N\}$.
- For each template S_k the set $\mathcal{P}(S_k)$ of possible critical paths is described.

1. Root vertex: $S := S_0$. Current set of paths: $Q = \emptyset$.
2. **Step 2 is this one!** For each $P \in \mathcal{P}(S)$:
   1. $Q := Q \cup \{P\}$.
   2. Find a critical instance $\tilde{I} = \arg\max_{I} \min_{P \in Q} P(I)$.
   3. If $\min_{P \in Q} P(\tilde{I}) \leqslant \rho^*$, $Q := Q \setminus \{P\}$, continue to the next P, if any.
      Otherwise climb up to the previous vertex, continue to the next P.
   4. Else
      1. Find an improving template $S \in \mathcal{S}$ for the instance $\tilde{I}$, Go to Step 2.
      2. If improving template is not found, STOP, output the critical instance.
3. Done!

Input files: model description

```
{   "M": 2,
    "J": 3,
    "params": 0,
    "missed_vertexes": "",
    "bounds": [
        [ "Lmax", "<", "1", "Lmax" ],
        [ "Dmax", "<", "1", "Dmax" ]
    ],
    "expressions": [],
    "hypothesis": "1",
    "objective_augmentation": "",
    "improvement_flag": true,
    "Machines": [
        [ 1, 2 ]
    ],
    "Jobs": [
        [ 1, 2, 3 ]
    ]}
```

Input files: template description

```
{
  "base": [
    [
      "a1 a2 a3",
      "b2 b3 b1",
      "a1 b1",
      "b2 a2",
      "b3 a3"
    ]
  ]
}
```

Input files: template description

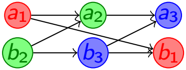
```
{
  "base": [
    [
      "a1 a2 a3",
      "b2 b3 b1",
      "a1 b1",
      "b2 a2",
      "b3 a3"
    ]
  ]
}
```

$$a_1 \to a_2 \to a_3,$$
$$b_2 \to b_3 \to b_1,$$
$$a_1 \to b_1,$$
$$b_2 \to a_2,$$
$$b_3 \to a_3.$$

Input files: template description

```
{
  "base": [
    [
      "a1 a2 a3",
      "b2 b3 b1",
      "a1 b1",
      "b2 a2",
      "b3 a3"
    ]
  ]
}
```

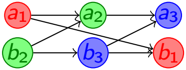
$$a_1 \to a_2 \to a_3,$$
$$b_2 \to b_3 \to b_1,$$
$$a_1 \to b_1,$$
$$b_2 \to a_2,$$
$$b_3 \to a_3.$$



Input files: template description

```
{
  "base": [
    [
      "a1 a2 a3",
      "b2 b3 b1",
      "a1 b1",
      "b2 a2",
      "b3 a3"
    ]
  ]
}
```

$$a_1 \to a_2 \to a_3,$$
$$b_2 \to b_3 \to b_1,$$
$$a_1 \to b_1,$$
$$b_2 \to a_2,$$
$$b_3 \to a_3.$$



$$\begin{cases} P_0 : b_2 + a_2 + a_3, \\ P_1 : b_2 + b_3 + b_1 \end{cases}$$

Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

Output: the tree of proof

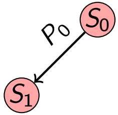
```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$

Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

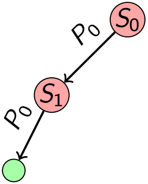
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

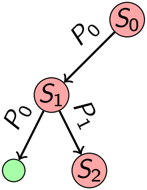
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

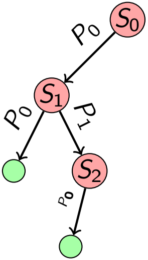
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

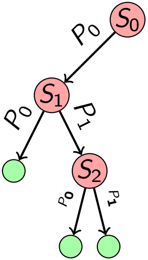
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

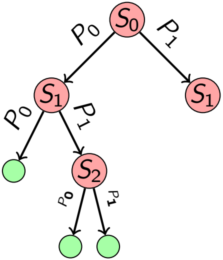
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=
    1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_
    2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1
    /2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1
    /2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_
    2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1
    /2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_
    3=1/2 ]
```

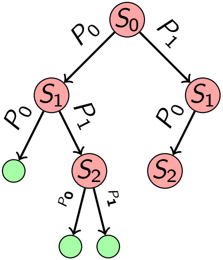
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

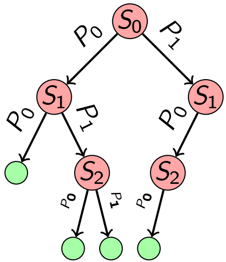
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

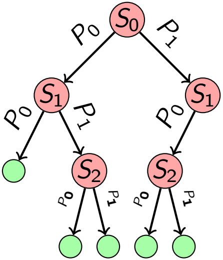
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

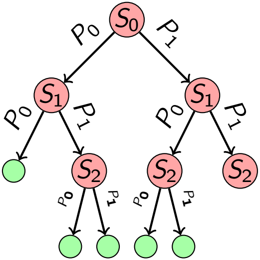
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

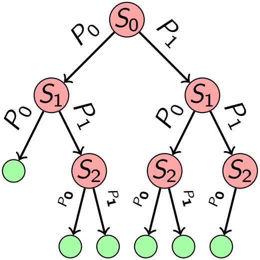
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=
   1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_
   2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1
   /2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1
   /2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_
   2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1
   /2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_
   3=1/2 ]
```

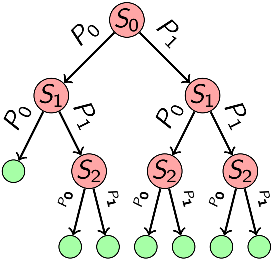
$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Output: the tree of proof

```
0 (0 1 2 )
0 0 [L1=1/2 L2=1/2 P_1=1/2 P_2=1/2 ]
0 1 (0 0 4 )
0 1 0 [L1=2/3 L2=1/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
0 1 1 [L1=1/2 L2=1/2 D3=0 P_2=1/2 P_3=1/2 ]
1 (0 1 2 )
1 0 (0 0 4 )
1 0 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 0 1 [L1=1/3 L2=2/3 P_1=1/3 P_2=1/3 P_3=1/3 ]
1 1 (0 0 4 )
1 1 0 [L1=1/2 L2=1/2 D3=0 P_1=1/2 P_3=1/2 ]
1 1 1 [L1=1/2 L2=1/2 P_2=1/2 P_3=1/2 ]
```

$$S_0 = S_{(0,0,0)}$$
$$S_1 = S_{(0,1,2)}$$
$$S_2 = S_{(0,0,4)}$$



Selected results

Problem	**Optima localization**
$O3\|\|C_{\max}$	$[SLB, 4/3SLB]$
$O3\|\nu = 2\|C_{\max}$	$[SLB, 5/4SLB]$
$F3\|pmtn\|C_{\max}$	$[SLB, 9/5SLB]$
$F4\|prmu, n \leq 4\|C_{\max}$	$[SLB, 13/6SLB]$

Selected results

Problem	**Optima localization**
$O3\|\|C_{\max}$	$[SLB, 4/3SLB]$
$O3\|\nu = 2\|C_{\max}$	$[SLB, 5/4SLB]$
$F3\|pmtn\|C_{\max}$	$[SLB, 9/5SLB]$
$F4\|prmu, n \leq 4\|C_{\max}$	$[SLB, 13/6SLB]$

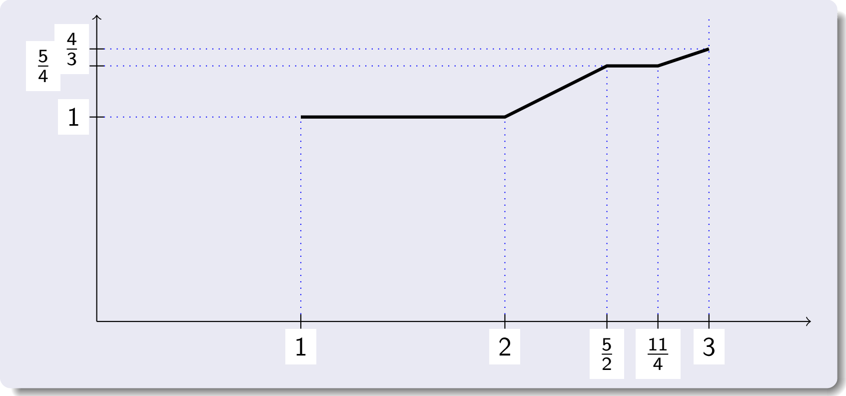
- It is proved, that for any instance I of the $O||C_{\max}$ problem $C^*_{\max}(I) \leqslant \Delta(I)/2$, where $\Delta(I) \doteq \sum \ell_i$ — total instance load.

Selected results

Problem	Optima localization
$O3\|\|C_{\max}$	$[SLB, 4/3SLB]$
$O3\|\nu = 2\|C_{\max}$	$[SLB, 5/4SLB]$
$F3\|pmtn\|C_{\max}$	$[SLB, 9/5SLB]$
$F4\|prmu, n \leq 4\|C_{\max}$	$[SLB, 13/6SLB]$

- It is proved, that for any instance I of the $O||C_{\max}$ problem $C^*_{\max}(I) \leqslant \Delta(I)/2$, where $\Delta(I) \doteq \sum \ell_i$ — total instance load.
- Exact form of the functional dependancy of the upper bound of optima localization interval on the total instance load for $O3||C_{\max}$.

- Exact form of the functional dependancy of the upper bound of optima localization interval on the total instance load for $O3||C_{\max}$



Open questions

- Optima localization for $O4||C_{\max}$ (it is unknown, if an instance I with $C^*_{\max}(I) > \frac{4}{3}SLB$ exists)

Open questions

- Optima localization for $O4||C_{\max}$ (it is unknown, if an instance I with $C^*_{\max}(I) > \frac{4}{3}SLB$ exists)
- Optima localization for $F4|prmu|C_{\max}$ (conjecture: $[SLB, \frac{13}{6}SLB]$)

Open questions

- Optima localization for $O4||C_{\max}$ (it is unknown, if an instance I with $C^*_{\max}(I) > \frac{4}{3}SLB$ exists)
- Optima localization for $F4|prmu|C_{\max}$ (conjecture: $[SLB, \frac{13}{6}SLB]$)
- Optima localization for $F4||C_{\max}$ (it is known, that upper bound of the interval is at least $\frac{67}{32}$)

Open questions

- Optima localization for $O4||C_{\max}$ (it is unknown, if an instance I with $C^*_{\max}(I) > \frac{4}{3}SLB$ exists)
- Optima localization for $F4|prmu|C_{\max}$ (conjecture: $[SLB, \frac{13}{6}SLB]$)
- Optima localization for $F4||C_{\max}$ (it is known, that upper bound of the interval is at least $\frac{67}{32}$)
- Sviridenko's conjecture: for any instance I of the $O||C_{\max}$ problem $C^*_{\max}(I) \leqslant SLB + p_{\max}$.

Thanks!

Thank you for attention!

Thanks!

Thank you for attention!

Questions?