Toward a Global Constraint for Minimizing the Flowtime

Camille Bonnin* Arnaud Malapert° Margaux Nattaf*
Marie-Laure Espinouse*

*Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
{camille.bonnin, margaux.nattaf, marie-laure.espinouse}@grenoble-inp.fr

°Université Côte d'Azur, CNRS, I3S, France

arnaud.malapert@univ-cotedazur.fr

13th International Conference on Operations Research and Enterprise Systems (ICORES 2024) 24-26 February 2024, Rome, Italy

Content

Context

Relaxations of $1|r_j, d_j; prec|\sum C_j$

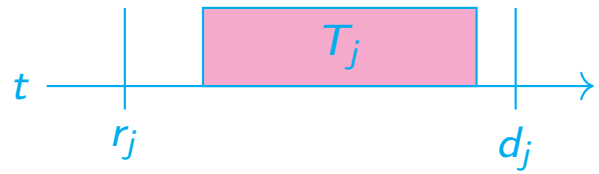
The FLOWTIME constraint

Experimental results

Conclusion and prospects

Notations:
machine | constraints | objectif

- 1: single-machine
- r_j: release dates
- d_j: deadlines
- $prec$: precedences
- $\sum C_j$: flowtime

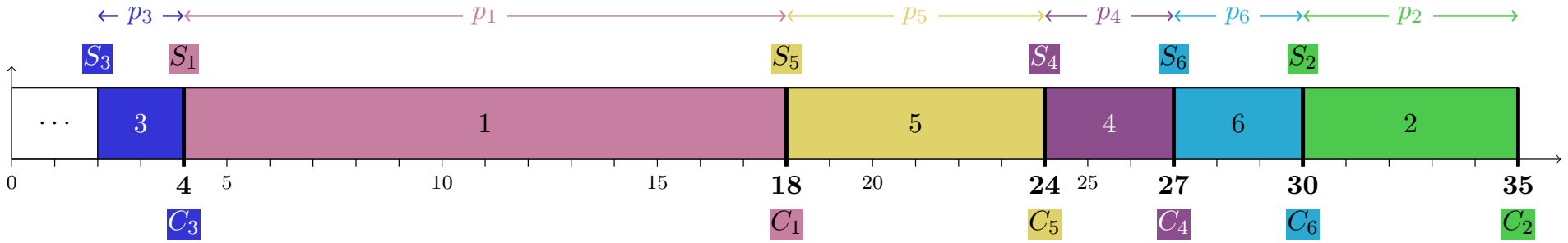


1. Context

Scheduling problems

Definition (scheduling)

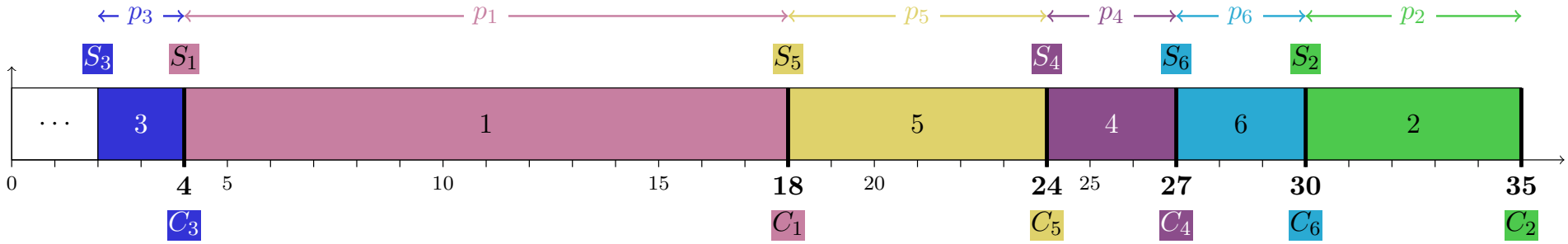
Allocating **resources** to **tasks** over **time** while respecting the tasks and resources **constraints**.
Often optimizing one or more **objectives** ($C_{\max}$, $\sum C_j \ldots$).



Scheduling problems

Definition (scheduling)

Allocating **resources** to **tasks** over **time** while respecting the tasks and resources **constraints**.
Often optimizing one or more **objectives** ($C_{\max}$, $\sum C_j \dots$).



Solving scheduling problems with CP

- Disjunctive constraint [Carlier, 1982; Fahimi & Quimper, 2014]
- Cumulative constraint [Aggoun & Beldiceanu, 1993; Guy et al., 2015]
- Survey of CP techniques for scheduling [Baptiste et al., 2001]
- CP vs MIP for 12 scheduling problems [Naderi et al., 2023]...

Constraint Optimization Problem (COP)

CP for scheduling: key points

- Declarative
- Global constraints
- Components for scheduling
- Exact method

Example in OPL, modeling language of IBM

```
dvar interval tasks[j in N] in r[j]..d[j] size p[j];
dvar int F; \\objective variable

minimize F;
subject to {
   noOverlap(tasks); \\resource constraint (ct)
   F = sum(j in N) endOf(tasks[j]); \\standard sum ct
   flowtime(tasks, F); \\contribution of this work
}
```

Constraint Programming (CP)

Principle

CP = **Filtering** + **Search**

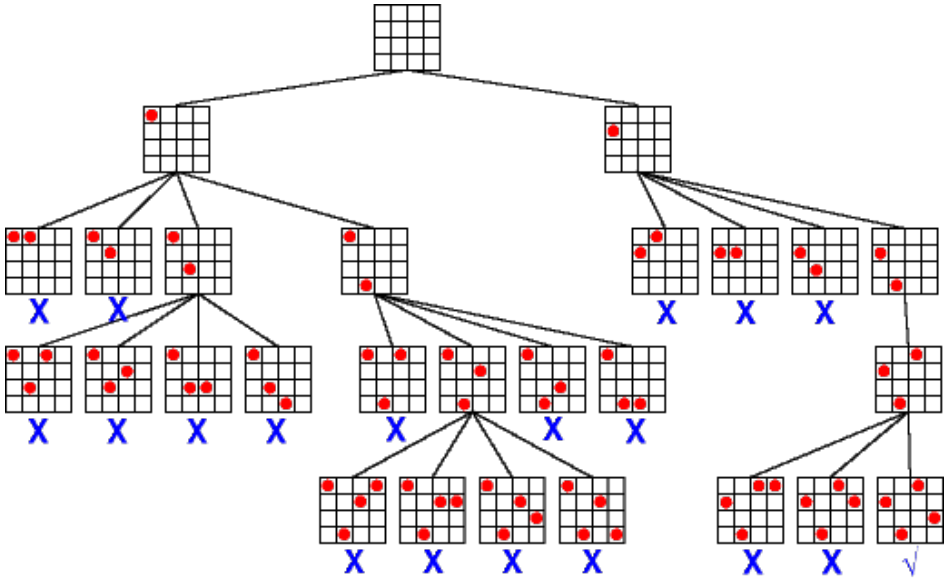


Figure: Solving the 4-queens problem with CP. Picture from the "Guide to Constraint Programming" of Roman Barták, 1998

Constraint Programming (CP)

Principle

CP = **Filtering** + **Search**

- **Filtering** = removing values from domains leading to **non feasible** solutions

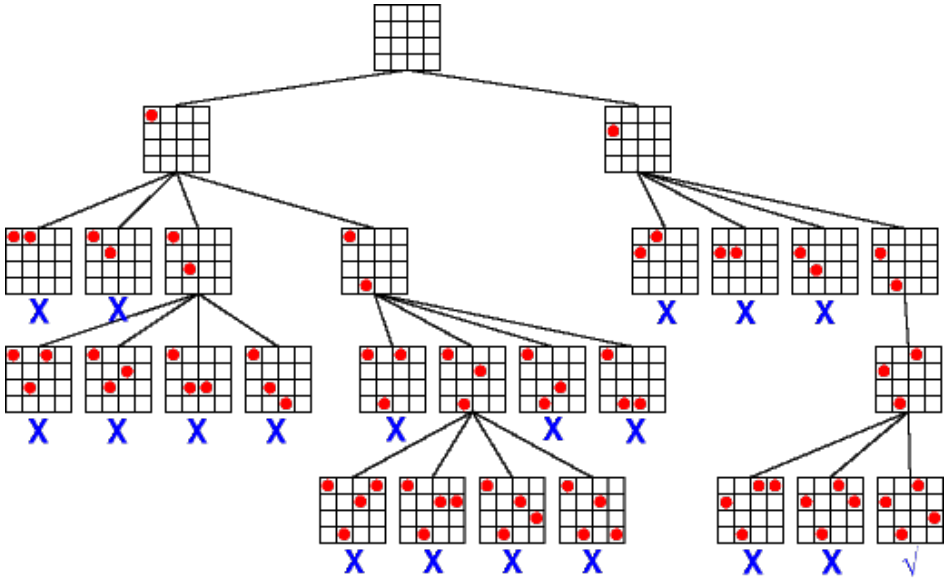


Figure: Solving the 4-queens problem with CP. Picture from the "Guide to Constraint Programming" of Roman Barták, 1998

Constraint Programming (CP)

Principle

CP = **Filtering** + **Search**

- **Filtering** = removing values from domains leading to **non feasible** solutions
- **Search** = **systematic** exploration of solution space

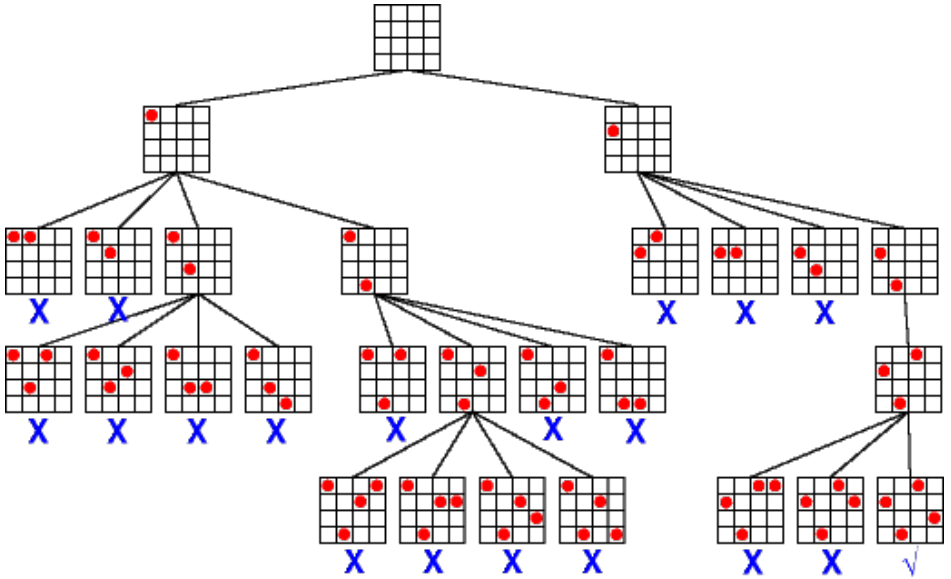


Figure: Solving the 4-queens problem with CP. Picture from the "Guide to Constraint Programming" of Roman Barták, 1998

Main objective and chosen approach

Thesis main objective

- Improve CP integration of **scheduling** objectives that are not studied a lot in CP, in particular "$\sum$" objectives

Main objective and chosen approach

Thesis main objective

- Improve CP integration of **scheduling** objectives that are not studied a lot in CP, in particular "$\sum$" objectives
- Focus on $\sum C_j$ (**flowtime**)

Main objective and chosen approach

Thesis main objective

- Improve CP integration of **scheduling** objectives that are not studied a lot in CP, in particular "$\sum$" objectives
- Focus on $\sum C_j$ (**flowtime**)

Chosen approach

- **Cost-based domain filtering [Focacci et al., 99]: global constraints** filter values not leading to a better-cost solution

Main objective and chosen approach

Thesis main objective

- Improve CP integration of **scheduling** objectives that are not studied a lot in CP, in particular "$\sum$" objectives
- Focus on $\sum C_j$ (**flowtime**)

Chosen approach

- **Cost-based domain filtering [Focacci et al., 99]: global constraints** filter values not leading to a better-cost solution
- **Scheduling problems:** a **relaxation** gives a **lower bound**

Main objective and chosen approach

Thesis main objective

- Improve CP integration of **scheduling** objectives that are not studied a lot in CP, in particular "$\sum$" objectives
- Focus on $\sum C_j$ (**flowtime**)

Chosen approach

- **Cost-based domain filtering [Focacci et al., 99]: global constraints** filter values not leading to a better-cost solution
- **Scheduling problems:** a **relaxation** gives a **lower bound**

Main objective of this work

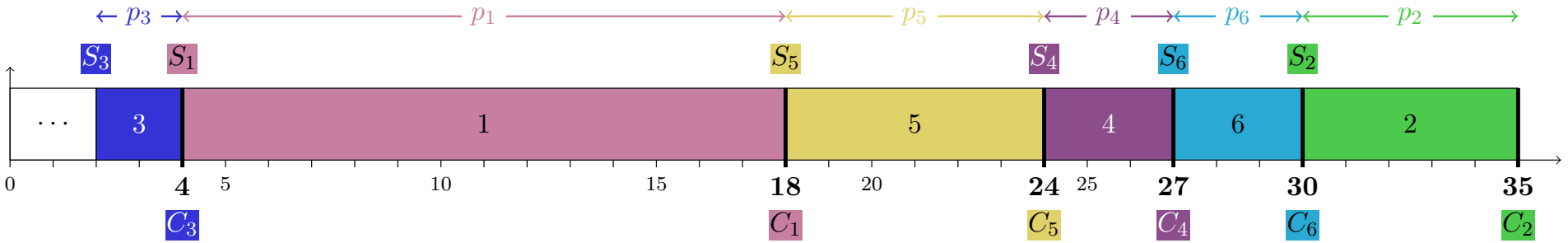
Propose a **new global constraint** helping minimizing $\sum C_j$ using **new polynomial relaxations**

The COMPLETION constraint

The COMPLETION constraint [Kovács & Beck, 2011]

$$\text{COMPLETION}([S_1, \ldots, S_n], [p_1, \ldots, p_n], [w_1, \ldots, w_n], C)$$

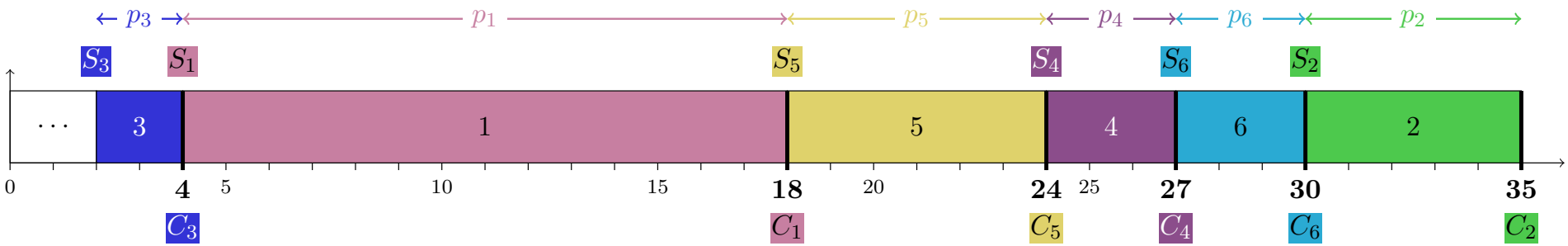
$$\overset{\text{def}}{\iff} ((\underbrace{S_i + p_i \leq S_j \vee S_j + p_j \leq S_i}_{\text{no tasks overlap}}) \wedge (\underbrace{\sum w_i(S_i + p_i) = C}_{\text{no preemption}}))$$



The COMPLETION constraint

The COMPLETION constraint [Kovács & Beck, 2011]

$$\textbf{COMPLETION}([S_1, \ldots, S_n], [p_1, \ldots, p_n], [w_1, \ldots, w_n], C)$$
$$\overset{\text{def}}{\iff} \underbrace{((S_i + p_i \leq S_j \vee S_j + p_j \leq S_i)}_{\text{no tasks overlap}} \wedge \underbrace{(\sum w_i(S_i + p_i) = C)}_{\text{no preemption}})$$

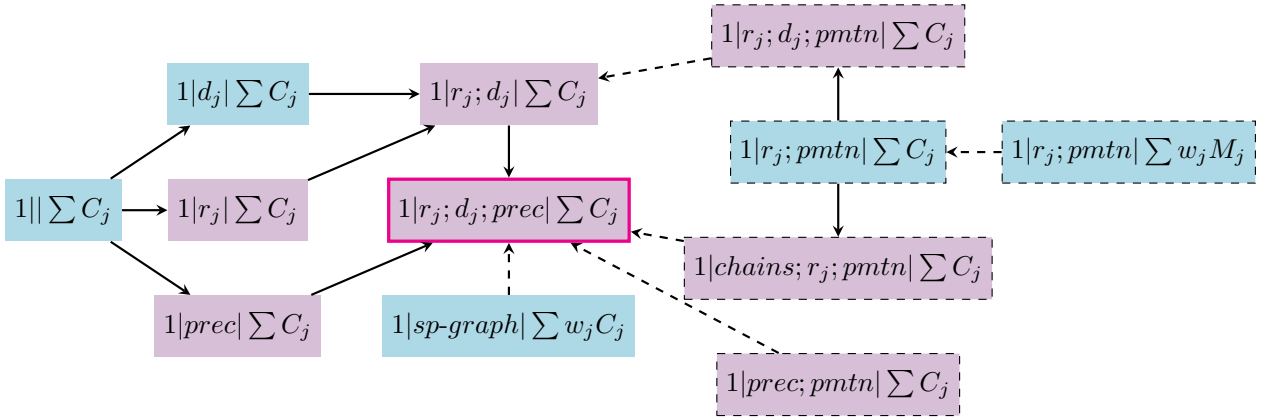


Main contribution of this work

Definition of a more efficient **global constraint** for the non-weighted case: **FLOWTIME**

2. Relaxations of $1|r_j, d_j; prec|\sum C_j$

Polynomial relaxations



$\mathcal{P}$

$\mathcal{NP}$-hard

problem	complexity	list algorithm	updated	reference
$1\|\|\sum C_j$	$O(n \log n)$	yes	no	[Horn, 1973]
$1\|d_j\|\sum C_j$	$O(n \log n)$	yes	yes	[Chen & al, 1998]
$1\|r_j;pmtn\|\sum C_j$	$O(n \log n)$	yes	yes	[Baker, 1974]
$1\|r_j;pmtn\|\sum w_jM_j$	$O(n \log n)$	yes	yes	[Kovàcs & Beck, 2011]
$1\|sp\text{-}graph\|\sum w_jC_j$	$O(n \log n)$	no	yes	[Lawler, 1978]

3. The FLOWTIME constraint

The FLOWTIME constraint

The COMPLETION constraint [Kovács & Beck, 2011]

- **COMPLETION**$([S_1, \ldots, S_n], [p_1, \ldots, p_n], [w_1, \ldots, w_n], \boldsymbol{C})$
$\overset{\text{def}}{\iff} ((\underbrace{S_i + p_i \leq S_j \vee S_j + p_j \leq S_i}_{\text{no tasks overlap}}) \wedge \underbrace{\sum w_i(S_i + p_i) = \boldsymbol{C}}_{\text{no preeemption}})$

The FLOWTIME constraint

- **FLOWTIME**$([S_1, \ldots, S_n], [p_1, \ldots, p_n], \boldsymbol{F})$
$\overset{\text{def}}{\iff} ((\underbrace{S_i + p_i \leq S_j \vee S_j + p_j \leq S_i}_{\text{no tasks overlap}}) \wedge \underbrace{\sum (S_i + p_i) = \boldsymbol{F}}_{\text{no preeemption}})$

The FLOWTIME constraint

The COMPLETION constraint [Kovács & Beck, 2011]

- **COMPLETION**$([S_1, \ldots, S_n], [p_1, \ldots, p_n], [w_1, \ldots, w_n], \boldsymbol{C})$
  $\overset{\text{def}}{\iff} ((\underbrace{S_i + p_i \leq S_j \vee S_j + p_j \leq S_i}_{\text{no tasks overlap}}) \wedge \underbrace{\sum w_i(S_i + p_i) = \boldsymbol{C}}_{\text{no preeemption}})$
- **Relaxation used:** $1|r_j; pmtn|\sum w_i M_i$

The FLOWTIME constraint

- **FLOWTIME**$([S_1, \ldots, S_n], [p_1, \ldots, p_n], \boldsymbol{F})$
  $\overset{\text{def}}{\iff} ((\underbrace{S_i + p_i \leq S_j \vee S_j + p_j \leq S_i}_{\text{no tasks overlap}}) \wedge \underbrace{\sum (S_i + p_i) = \boldsymbol{F}}_{\text{no preeemption}})$
- **Relaxation used:** many relaxations possible

Global schema

Global schema of the filtering algorithm

- Choose a relaxation

Global schema

Global schema of the filtering algorithm

- **Choose a relaxation**
- Two rules:

Global schema

Global schema of the filtering algorithm

- **Choose a relaxation**
- Two rules:
  - **Update the lower bound** of the objective variable ($\underline{F}$)

Global schema

Global schema of the filtering algorithm

- **Choose a relaxation**
- Two rules:
  - **Update the lower bound** of the objective variable ($\underline{F}$)
  - **Filter the bounds** of the domains of the S_j variables

Global schema

Global schema of the filtering algorithm

- **Choose a relaxation**
- Two rules:
  - **Update the lower bound** of the objective variable ($\underline{F}$)
  - **Filter the bounds** of the domains of the S_j variables

Remarks

- The relaxation can be **changed**

Global schema

Global schema of the filtering algorithm

- **Choose a relaxation**
- Two rules:
  - **Update the lower bound** of the objective variable ($\underline{F}$)
  - **Filter the bounds** of the domains of the S_j variables

Remarks

- The relaxation can be **changed**
- The relaxation is considered a **"black box"**

Global schema

Global schema of the filtering algorithm

- **Choose a relaxation**
- Two rules:
  - **Update the lower bound** of the objective variable ($\underline{F}$)
  - **Filter the bounds** of the domains of the S_j variables

Remarks

- The relaxation can be **changed**
- The relaxation is considered a **"black box"**
- Allows to **compare the performances** of different relaxations

Objective bound update: example

T_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	1	12	16	17
d_j	24	∞	10	∞	26	∞

Objective bound update: example

T_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	1	12	16	17
d_j	24	∞	10	∞	26	∞

Relaxation used: $1|r_j; pmtn|\sum C_j$

Objective bound update: example

T_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	1	12	16	17
d_j	24	∞	10	∞	26	∞

Relaxation used: $1|r_j; pmtn|\sum C_j$

$F \in [100, 130]$

j	1	2	3	4	5	6
S_j	[0, 10]	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	0	0	1	12	16	17
d'_j	24	51	10	51	26	20

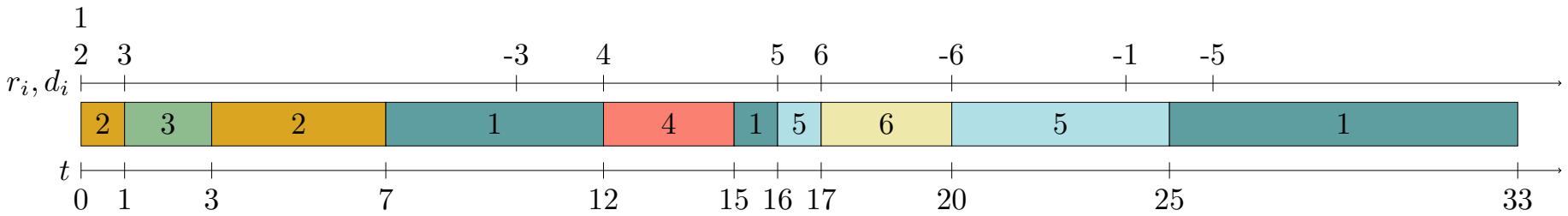
Objective bound update: example

T_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	1	12	16	17
d_j	24	∞	10	∞	26	∞

Relaxation used: $1|r_j; pmtn|\sum C_j$

$F \in [100, 130]$

j	1	2	3	4	5	6
S_j	[0, 10]	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	0	0	1	12	16	17
d'_j	24	51	10	51	26	20



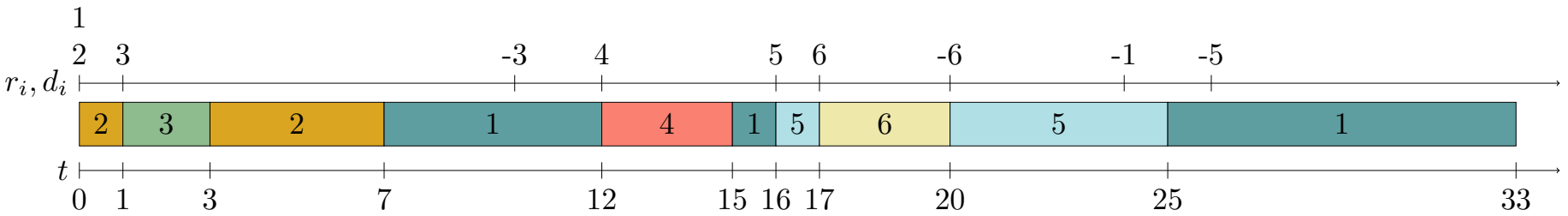
Objective bound update: example

T_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	1	12	16	17
d_j	24	∞	10	∞	26	∞

Relaxation used: $1|r_j; pmtn|\sum C_j$

$F \in [100, 130]$ $\quad \sum C_j = \mathbf{104} > \underline{F}$

j	1	2	3	4	5	6
S_j	[0, 10]	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	0	0	1	12	16	17
d'_j	24	51	10	51	26	20



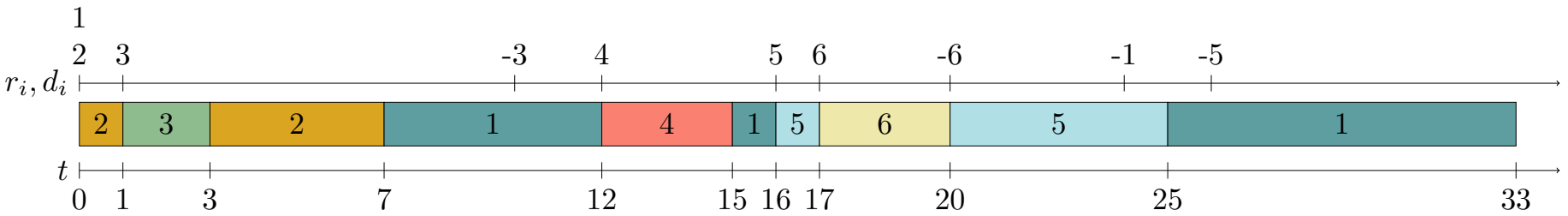
Objective bound update: example

T_j	1	2	3	4	5	6
p_j	14	5	2	3	6	3
r_j	0	0	1	12	16	17
d_j	24	∞	10	∞	26	∞

Relaxation used: $1|r_j; pmtn|\sum C_j$

$F \in [\mathbf{104}, 130]$ $\quad \sum C_j = \mathbf{104} > \underline{F}$

j	1	2	3	4	5	6
S_j	[0, 10]	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	0	0	1	12	16	17
d'_j	24	51	10	51	26	20



Filter bounds of S_j: example

Filter $\underline{S_1}$ (symmetrical for $\overline{S_1}$)

$F \in [104, 130]$ Relaxation used: $1|r_j; pmtn|\sum C_j$

j	**1**	2	3	4	5	6
S_j	**[2, 10]**	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	**2**	0	1	12	16	17
d'_j	**24**	51	10	51	26	20

Filter bounds of S_j: example

Filter $\underline{S_1}$ (symmetrical for $\overline{S_1}$)

$F \in [104, 130]$ Relaxation used: $1|r_j; pmtn|\sum C_j$

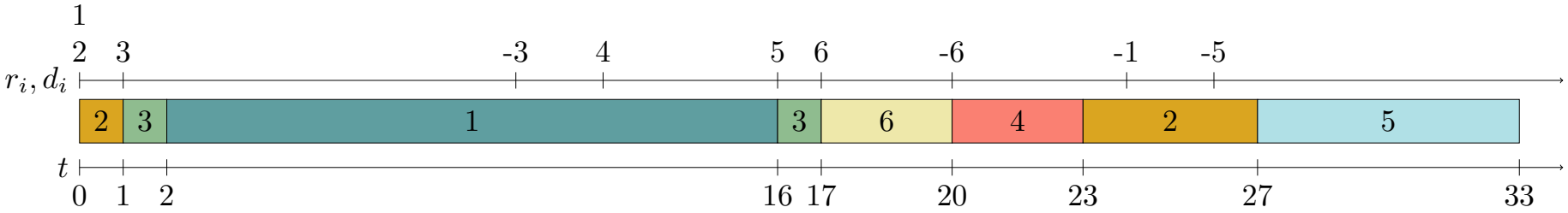
j	**1**	2	3	4	5	6
S_j	**[2, 10]**	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	**2**	0	1	12	16	17
d'_j	**24**	51	10	51	26	20

Filter bounds of S_j: example

Filter $\underline{S_1}$ (symmetrical for $\overline{S_1}$)

$F \in [104, 130]$ Relaxation used: $1|r_j; pmtn|\sum C_j$

j	**1**	2	3	4	5	6
S_j	**[2, 10]**	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	**2**	0	1	12	16	17
d'_j	**24**	51	10	51	26	20



Step 1 : $t = 2$, $relax_{1,2} = 136 > 130 (= \overline{F})$

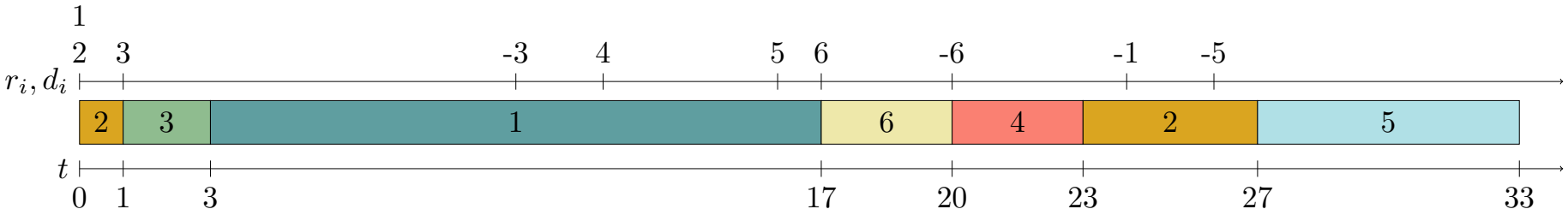
S_1: 2 is filtered $\Rightarrow$ $t = t + 1$

Filter bounds of S_j: example

Filter $\underline{S_1}$ (symmetrical for $\overline{S_1}$)

$F \in [104, 130]$ Relaxation used: $1|r_j; pmtn|\sum C_j$

j	**1**	2	3	4	5	6
S_j	**[3, 10]**	[0, 46]	[1, 8]	[12, 49]	[16, 20]	[17, 17]
r'_j	**3**	0	1	12	16	17
d'_j	**24**	51	10	51	26	20



Step 2 : $t = 3$, $relax_{1,3} = 123 \leq 130 (= \overline{F})$

STOP

4. Experimental results

Experimental environment

Environment and parameters

- C++ and IBM ILOG CPLEX Optimization Studio 22.1

Experimental environment

Environment and parameters

- C++ and **IBM ILOG CPLEX Optimization Studio 22.1**
- The code of **COMPLETION** has been **adapted** to the solver version (`IlcActivity` $\longrightarrow$ `IloIntervalVar`)

Experimental environment

Environment and parameters

- **C++** and **IBM ILOG CPLEX Optimization Studio 22.1**
- The code of **COMPLETION** has been **adapted** to the solver version (`IlcActivity` $\longrightarrow$ `IloIntervalVar`)
- Dell computer with **256 GB** of **RAM** and **4** Intel E7-4870 2.40 GHz **processors** running on **CentOS Linux** release 7.9 (each processor has 10 cores)

Experimental environment

Environment and parameters

- **C++** and **IBM ILOG CPLEX Optimization Studio 22.1**
- The code of **COMPLETION** has been **adapted** to the solver version (`IlcActivity` $\longrightarrow$ `IloIntervalVar`)
- Dell computer with **256 GB** of **RAM** and **4** Intel E7-4870 2.40 GHz **processors** running on **CentOS Linux** release 7.9 (each processor has 10 cores)
- **Parallelism** is **disabled**

Experimental environment

Environment and parameters

- **C++** and **IBM ILOG CPLEX Optimization Studio 22.1**
- The code of **COMPLETION** has been **adapted** to the solver version (`IlcActivity` $\longrightarrow$ `IloIntervalVar`)
- Dell computer with **256 GB** of **RAM** and **4** Intel E7-4870 2.40 GHz **processors** running on **CentOS Linux** release 7.9 (each processor has 10 cores)
- **Parallelism** is **disabled**
- Time limit is **1000 seconds** for each run

Instances

Instances used

- **Single-machine** problems: $1|r_j|\sum C_j$

Instances

Instances used

- **Single-machine** problems: $1|r_j|\sum C_j$
- **900 instances** from **[Pan & Shi, 08]**

Instances

Instances used

- **Single-machine** problems: $1|r_j|\sum C_j$
- **900 instances** from **[Pan & Shi, 08]**
- **Less than 100 tasks**

Instances

Instances used

- **Single-machine** problems: $1|r_j|\sum C_j$
- **900 instances** from **[Pan & Shi, 08]**
- **Less than 100 tasks**
- Different relative ranges of the release time

Alternatives tested

Main alternatives tested

- Most efficient **alternatives** for the **FLOWTIME** constraint:

name	filter S_j	relaxation
`pmtnFlow`	no	$1\|r_j; pmtn\|\sum C_j$
`pmtnBusy`	no	$1\|r_j; pmtn\|\sum M_j$
`filtFlow`	yes	$1\|r_j; pmtn\|\sum C_j$
`filtBusy`	yes	$1\|r_j; pmtn\|\sum M_j$

Alternatives tested

Main alternatives tested

- Most efficient **alternatives** for the **FLOWTIME** constraint:

name	filter S_j	relaxation
`pmtnFlow`	no	$1\|r_j; pmtn\|\sum C_j$
`pmtnBusy`	no	$1\|r_j; pmtn\|\sum M_j$
`filtFlow`	yes	$1\|r_j; pmtn\|\sum C_j$
`filtBusy`	yes	$1\|r_j; pmtn\|\sum M_j$

- `sum`: **standard sum** constraint

Alternatives tested

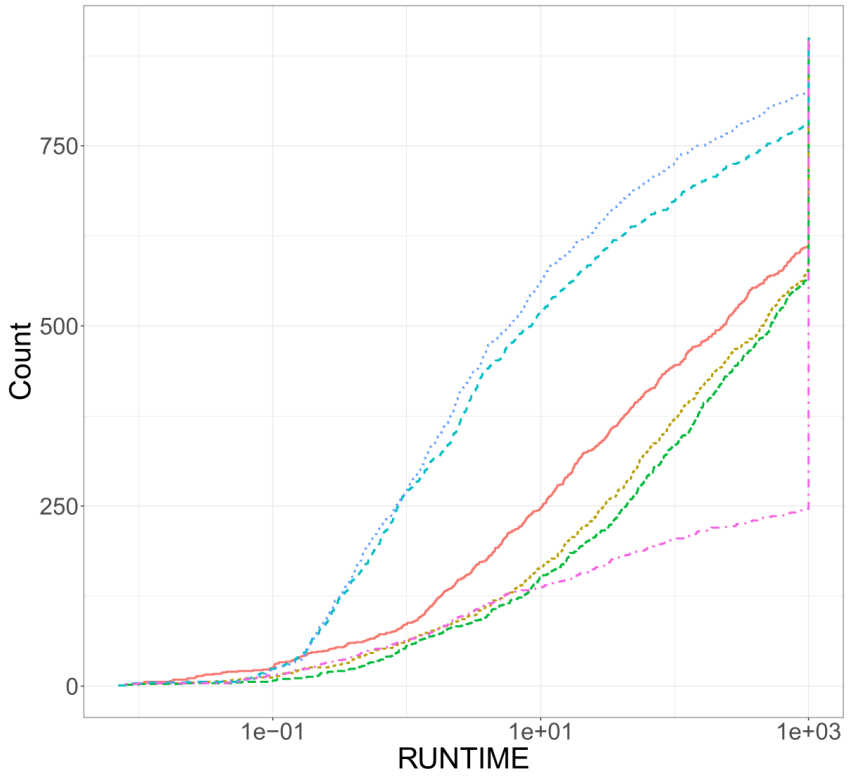
Main alternatives tested

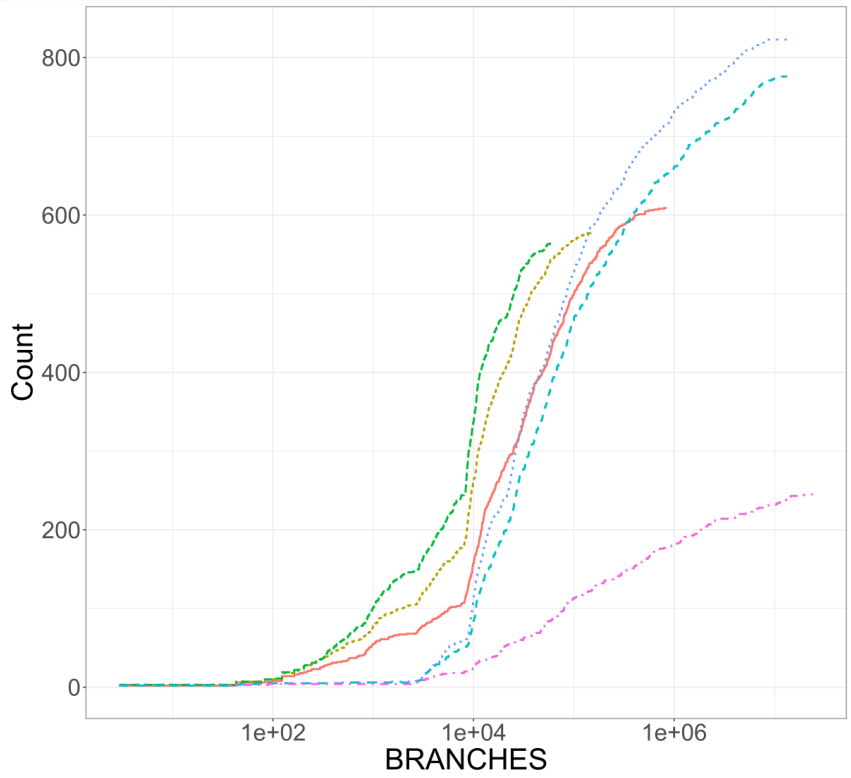
- Most efficient **alternatives** for the **FLOWTIME** constraint:

name	filter S_j	relaxation
`pmtnFlow`	no	$1\|r_j; pmtn\|\sum C_j$
`pmtnBusy`	no	$1\|r_j; pmtn\|\sum M_j$
`filtFlow`	yes	$1\|r_j; pmtn\|\sum C_j$
`filtBusy`	yes	$1\|r_j; pmtn\|\sum M_j$

- `sum`: **standard sum** constraint
- `complBusy`: adapts **COMPLETION** $(1|r_j; pmtn|\sum M_j)$

Experimental results





complBusy filtFlow pmtnFlow
filtBusy pmtnBusy sum

Time: pmtnFlow < pmtnBusy < complBusy < filtBusy < filtFlow < sum

Branches: filtFlow < filtBusy < complBusy < pmtnFlow < pmtnBusy < sum

Detailed results

alternative	OPT	SAT	mean t*	median t*	mean #b*	median #b*
pmtnFlow	**824**	76	**50.5**	**2.62**	484613	51618
pmtnBusy	777	123	55.4	2.87	666864	60633
filtFlow	565	335	169	58.8	**10776**	**8704**
filtBusy	578	322	158	47.3	18942	10832
sum	246	654	76.9	5.21	1629165	150508
complBusy	610	290	114	17.8	63835	25356

[*]only on OPT solved instances

Conclusion of experimental results

Summary of the results

- **Less time** taken to solve single-machine problems **with FLOWTIME**

Conclusion of experimental results

Summary of the results

- **Less time** taken to solve single-machine problems **with FLOWTIME**
- **+500** more instances solved to **optimality** with `pmtnFlow` than with `sum`

Conclusion of experimental results

Summary of the results

- **Less time** taken to solve single-machine problems **with FLOWTIME**
- **+500** more instances solved to **optimality** with `pmtnFlow` than with `sum`
- `pmtnFlow` **quicker** than `pmtnBusy`, and visits **less branches**

Conclusion of experimental results

Summary of the results

- **Less time** taken to solve single-machine problems **with FLOWTIME**
- **+500** more instances solved to **optimality** with `pmtnFlow` than with `sum`
- `pmtnFlow` **quicker** than `pmtnBusy`, and visits **less branches**
- `complBusy` visits **more branches** than `filtFlow`

Conclusion of experimental results

Summary of the results

- **Less time** taken to solve single-machine problems **with FLOWTIME**
- **+500** more instances solved to **optimality** with `pmtnFlow` than with `sum`
- `pmtnFlow` **quicker** than `pmtnBusy`, and visits **less branches**
- `complBusy` visits **more branches** than `filtFlow`
- Currently, **filtering** S_j: uses **less branches** than **just updating** $\underline{F}$, but takes **more time**

5. Conclusion and prospects

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$
- Filtering algorithm allows **using different relaxations**

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$
- Filtering algorithm allows **using different relaxations**
- **Effective** for **single-machine** problems

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$
- Filtering algorithm allows **using different relaxations**
- **Effective** for **single-machine** problems
- **Less effective** for **flowshop** problems

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$
- Filtering algorithm allows **using different relaxations**
- **Effective** for **single-machine** problems
- **Less effective** for **flowshop** problems

Prospects

- **Improve** the **incrementality** of the filtering algorithm ($t = t + 1$)

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$
- Filtering algorithm allows **using different relaxations**
- **Effective** for **single-machine** problems
- **Less effective** for **flowshop** problems

Prospects

- **Improve** the **incrementality** of the filtering algorithm ($t = t + 1$)
- **Add conditions** to use the filtering algorithm

Conclusion and prospects

Conclusion

- **Global constraint** minimizing $\sum C_j$
- FLOWTIME can be used to **model any problem with** $\sum C_j$
- Filtering algorithm allows **using different relaxations**
- **Effective** for **single-machine** problems
- **Less effective** for **flowshop** problems

Prospects

- **Improve** the **incrementality** of the filtering algorithm $(t = t + 1)$
- **Add conditions** to use the filtering algorithm
- **Add** $1|\textit{sp-graph}|\sum w_j C_j$ relaxation