



### Extending Smith's Rule with Task Mandatory Parts and Release Dates

### Bonnin Camille \*, Nattaf Margaux\* Malapert Arnaud<sup>o</sup> Espinouse Marie-Laure\*

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

<sup>o</sup>Université Côte d'Azur, CNRS, I3S, France

arnaud.malapert@univ-cotedazur.fr

### PMS 2022, April 6-7, Ghent



Laboratoire des Sciences pour la Conception. l'Optimisation et la production

Introduction

0000

G. SCOP

### 1. Introduction

Introduction Modified Smith Rule with MPs Experimentations Conclu





#### Note: here and in the following, we will use $d_j$ instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of  $\sum C_i$  objective.

- General problem (even without prec):  $1|r_j; d_j| \sum C_j$
- Polynomial relaxation with a list algorithm:  $1|r_j$ ; *pmtn* $|\sum C_j|$
- We want a Lower Bound (LB) of good quality and computable quickly

 $\sum C_j = 138$ (4 + 18 + 24 + 27 + 30 + 35)

GiSCOP



#### Note: here and in the following, we will use $d_j$ instead of $\widetilde{d}_j$ for deadlines

Long term objective: improve CP integration of  $\sum C_i$  objective

- General problem (even without prec):  $1|r_j; d_j| \sum C_j$
- Polynomial relaxation with a list algorithm:  $1|r_j$ ; pmtn $|\sum C_j|$
- We want a Lower Bound (LB) of good quality and computable quickly

 $\sum_{j=1}^{1} C_{j} = 138$ (4 + 18 + 24 + 27 + 30 + 35)

#### Note: here and in the following, we will use $d_j$ instead of $\tilde{d}_j$ for deadlines

### Long term objective: improve CP integration of $\sum C_i$ objective

- General problem (even without prec):  $1|r_j; d_j| \sum C_j$
- Polynomial relaxation with a list algorithm:  $1|r_i$ ; *pmtn*| $\sum C_i$
- We want a Lower Bound (LB) of good quality and computable quickly

$$\sum C_j = 138$$
(4 + 18 + 24 + 27 + 30 + 35)

#### Note: here and in the following, we will use $d_j$ instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of  $\sum C_j$  objective

- General problem (even without prec):  $1|r_j; d_j| \sum C_j$
- Polynomial relaxation with a list algorithm:  $1|r_j$ ; *pmtn*  $|\sum C_j|$
- We want a Lower Bound (LB) of good quality and computable quickly

2/21

 $\sum C_j = 138$ (4 + 18 + 24 + 27 + 30 + 35)

#### Note: here and in the following, we will use $d_j$ instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of  $\sum C_i$  objective

- General problem (even without prec):  $1|r_j; d_j| \sum C_j$
- Polynomial relaxation with a list algorithm:  $1|r_j$ ; *pmtn*  $|\sum C_j|$
- We want a Lower Bound (LB) of good quality and computable quickly

 $\sum C_j = 138$ (4 + 18 + 24 + 27 + 30 + 35)

#### Note: here and in the following, we will use $d_j$ instead of $\tilde{d}_j$ for deadlines

Long term objective: improve CP integration of  $\sum C_i$  objective

- General problem (even without prec):  $1|r_j; d_j| \sum C_j$
- Polynomial relaxation with a list algorithm:  $1|r_j$ ; *pmtn*  $|\sum C_j|$
- We want a Lower Bound (LB) of good quality and computable quickly

idj	1	2	3	4	5	6
<b>p</b> j	14	5	2	3	6	3
rj	0	0	2	12	16	18
dj	24	$\infty$	4	$\infty$	26	$\infty$

 $\sum C_j = 138$ (4 + 18 + 24 + 27 + 30 + 35)



### Modified Smith Rule with MPs Experior

### **Problem**

### Note: here and in the following, we will use $d_j$ instead of $ilde{d}_j$ for deadlines

### Long term objective: improve CP integration of $\sum C_j$ objective

• General problem (even without prec):  $1|r_j; d_j| \sum C_j$ 

Introduction

- Polynomial relaxation with a list algorithm:  $1|r_j$ ; *pmtn* $|\sum C_j$
- We want a Lower Bound (LB) of good quality and computable quickly





### Introduction



### Note: here and in the following, we will use $d_i$ instead of $\tilde{d}_i$ for deadlines

### <u>Long term objective</u>: improve CP integration of $\sum C_i$ objective

• General problem (even without prec):  $1|r_i; d_i| \sum C_i$ 

0000

- Polynomial relaxation with a list algorithm:  $1|r_i$ ; pmtn  $\sum C_i$
- We want a Lower Bound (LB) of good quality and computable quickly





### Introduction

### **Problem**

### Note: here and in the following, we will use $d_i$ instead of $\tilde{d}_i$ for deadlines

### <u>Long term objective</u>: improve CP integration of $\sum C_i$ objective

• General problem (even without prec):  $1|r_i; d_i| \sum C_i$ 

0000

- Polynomial relaxation with a list algorithm:  $1|r_i$ ; pmtn  $\sum C_i$
- We want a Lower Bound (LB) of good quality and computable quickly



Introduction



### Literature

G-SCOP

•  $1||\sum C_i$ : Smith Rule (or SPT rule) [Smith, 1956]

0000

•  $1|r_i; pmtn| \sum C_i$ : Modified Smith Rule (MSR) [Baker, 1974]

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of  $1|r_i; d_i| \sum C_i$  respecting MPs
- Long term objective: improve CP solving of  $1|r_i; d_i| \sum C_i$

Introduction Modified Smith Rule with MPs Experimentations Conclusi



### Literature

G-SCOP

- $1||\sum C_j$ : Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn| \sum C_j$ : Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006]  $\Rightarrow 1^{st}$  Lower Bound (LB)

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of  $1|r_j; d_j| \sum C_j$  respecting MPs
- Long term objective: improve CP solving of  $1|r_j; d_j| \sum C_j$

### Literature

- $1||\sum C_j$ : Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn| \sum C_j$ : Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006]  $\Rightarrow 1^{st}$  Lower Bound (LB)

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of  $1|r_j; d_j| \sum C_j$  respecting MPs
- Long term objective: improve CP solving of  $1|r_j; d_j| \sum C_j$

GISCOP

### Literature

- $1||\sum C_j$ : Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn| \sum C_j$ : Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006]  $\Rightarrow 1^{st}$  Lower Bound (LB)

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of  $1|r_j; d_j| \sum C_j$  respecting MPs
- Long term objective: improve CP solving of  $1|r_j; d_j| \sum C_j$

### Literature

- $1||\sum C_j$ : Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn| \sum C_j$ : Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006]  $\Rightarrow 1^{st}$  Lower Bound (LB)

### Objectives

Extend MSR to enforce the tasks Mandatory Parts (MP)

• Compute a LB of  $1|r_j; d_j| \sum C_j$  respecting MPs

• Long term objective: improve CP solving of  $1|r_j; d_j| \sum C_j$ 

### Literature

- $1||\sum C_j$ : Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn| \sum C_j$ : Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006]  $\Rightarrow 1^{st}$  Lower Bound (LB)

### Objectives

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of  $1|r_j; d_j| \sum C_j$  respecting MPs

ullet Long term objective: improve CP solving of  $1|r_j;d_j|\sum C_j$ 

### Literature

- $1||\sum C_j$ : Smith Rule (or SPT rule) [Smith, 1956]
- $1|r_j; pmtn| \sum C_j$ : Modified Smith Rule (MSR) [Baker, 1974 and Brucker, 2006]  $\Rightarrow 1^{st}$  Lower Bound (LB)

- Extend MSR to enforce the tasks Mandatory Parts (MP)
- Compute a LB of  $1|r_j; d_j| \sum C_j$  respecting MPs
- Long term objective: improve CP solving of  $1|r_j; d_j| \sum C_j$

Introduction

Modified Smith Rule with

Experimentatio

### Mandatory Part (MP)

### Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

 $eft_j = earliest finishing time of task j$  $<math>lst_j = latest starting time of task j$ 

G-SCOP

Introduction

Modified Smith Rule with

Experimentatio

### Mandatory Part (MP)

### Definition: Mandatory Part of a task (MP)

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.

 $eft_j = earliest$  finishing time of task j $lst_j = latest$  starting time of task j

G-SCOP



### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.





### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.





### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



 $eft_j$  = earliest finishing time of task j $lst_j$  = latest starting time of task j



### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



 $eft_j$  = earliest finishing time of task j $lst_j$  = latest starting time of task j



### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



 $eft_j$  = earliest finishing time of task j $Ist_j$  = latest starting time of task j



### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



 $eft_j$  = earliest finishing time of task j $lst_j$  = latest starting time of task j



### Definition: Mandatory Part of a task (MP)

Introduction

Time interval in which a task **must be executed** under the hypothesis that it is **not late**, **not preempted** and without taking into account the other tasks.



 $eft_j$  = earliest finishing time of task j $Ist_j$  = latest starting time of task j



ntroduction 0000 Modified Smith Rule with MPs

Conclusion

### 2. Modified Smith Rule with MPs



### 

Modified Smith Rule with MPs

### Modified Smith Rule (MSR)

### Theorem: Modified Smith Rule (MSR) [Brucker, 2006]

At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .

# $\sum_{i} C_{j} = 105$ 4 + 7 + 15 + 21 + 25 + 33

**Complexity**:  $O(n \log n)$  with a list algorithm



At each release time or finishing time of a task, schedule an unfinished task which is available and has the smallest remaining processing time  $(\hat{p}_j)$ .

$$\sum C_j = 105$$
4 + 7 + 15 + 21 + 25 + 33)

**Complexity**:  $O(n \log n)$  with a list algorithm

6/21



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: *O*(*n* log *n*) with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: *O*(*n* log *n*) with a list algorithm


At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_i)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each release time or finishing time of a task, schedule an unfinished task which is available and has the smallest remaining processing time  $(\hat{p}_j)$ .



### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: O(n log n) with a list algorithm



At each release time or finishing time of a task, schedule an unfinished task which is available and has the smallest remaining processing time  $(\hat{p}_j)$ .



### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_i)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_i)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_i)$ .



#### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_i)$ .



### **Complexity**: $O(n \log n)$ with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm



At each **release time** or **finishing time** of a task, schedule an unfinished task which is available and has the **smallest remaining processing time**  $(\hat{p}_j)$ .



**Complexity**:  $O(n \log n)$  with a list algorithm

6/21



7/21

# Modified Smith Rule with MPs

Modified Smith Rule with MPs

800000000

# Theorem: Modified Smith Rule with MPs

- Schedule MPs on their intervals
- Ose MSR on the tasks without their MP
  - In case of equality, schedule the unfinished task without MP or with the earliest MP

 $\sum C_j = 110$  (4 + 7 + 17 + 23 + 26 + 33)Complexity:  $O(n \log n)$  with a sweep line algorithm



 In case of equality, schedule the unfinished task without MP or with the earliest MP



#### **Complexity**: $O(n \log n)$ with a sweep line algorithm



























or with the earliest MI



#### **Complexity**: $O(n \log n)$ with a sweep line algorithm



or with the earliest MP



#### **Complexity**: $O(n \log n)$ with a sweep line algorithm



 In case of equality, schedule the unfinished task without MP or with the earliest MP



#### **Complexity**: $O(n \log n)$ with a sweep line algorithm



 In case of equality, schedule the unfinished task without MP or with the earliest MP



#### **Complexity**: $O(n \log n)$ with a sweep line algorithm




### **Complexity**: $O(n \log n)$ with a sweep line algorithm



or with the earliest MI



**Complexity**:  $O(n \log n)$  with a sweep line algorithm





### **Complexity**: $O(n \log n)$ with a sweep line algorithm





#### **Complexity**: $O(n \log n)$ with a sweep line algorithm



or with the earliest MF



**Complexity**:  $O(n \log n)$  with a sweep line algorithm





#### **Complexity**: $O(n \log n)$ with a sweep line algorithm





#### **Complexity**: $O(n \log n)$ with a sweep line algorithm





#### **Complexity**: $O(n \log n)$ with a sweep line algorithm





### **Complexity**: $O(n \log n)$ with a sweep line algorithm





#### **Complexity**: $O(n \log n)$ with a sweep line algorithm





#### **Complexity**: $O(n \log n)$ with a sweep line algorithm





### **Complexity**: $O(n \log n)$ with a sweep line algorithm































7/21

# Modified Smith Rule with MPs

### Theorem: Modified Smith Rule with MPs

- O Schedule MPs on their intervals
- **2** Use **MSR** on the tasks without their MP
  - In case of equality, schedule the unfinished task without MP or with the earliest MP

Modified Smith Rule with MPs

800000000



**Complexity**: *O*(*n* log *n*) with a sweep line algorithm



7/21

# Modified Smith Rule with MPs

### Theorem: Modified Smith Rule with MPs

- O Schedule MPs on their intervals
- **2** Use **MSR** on the tasks without their MP
  - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

Modified Smith Rule with MPs

800000000



**Complexity**:  $O(n \log n)$  with a sweep line algorithm



# Modified Smith Rule with MPs

### Theorem: Modified Smith Rule with MPs

- O Schedule MPs on their intervals
- **2** Use **MSR** on the tasks without their MP
  - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

Modified Smith Rule with MPs

800000000



**Complexity**:  $O(n \log n)$  with a sweep line algorithm



7/21

# Modified Smith Rule with MPs

### Theorem: Modified Smith Rule with MPs

- O Schedule MPs on their intervals
- **2** Use **MSR** on the tasks without their MP
  - In case of **equality**, schedule the unfinished task **without MP** or with the **earliest MP**

Modified Smith Rule with MPs

800000000



**Complexity**:  $O(n \log n)$  with a sweep line algorithm



# Modified Smith Rule with MPs

### Theorem: Modified Smith Rule with MPs

- O Schedule MPs on their intervals
- **2** Use **MSR** on the tasks without their MP
  - In case of equality, schedule the unfinished task without MP or with the earliest MP

Modified Smith Rule with MPs

800000000



**Complexity**:  $O(n \log n)$  with a sweep line algorithm



Modified Smith Rule with MPs

800000000

## Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_i$ ; *pmtn*;  $MP|\sum C_i$ .

- S = solution of MSR with MPs;
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S



Modified Smith Rule with MPs

000000000

## Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ;
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S<sup>\*</sup>



Modified Smith Rule with MPs

800000000

## Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than J
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j => S<sup>\*</sup>
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S



Modified Smith Rule with MPs

800000000

# Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than J
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j => S<sup>\*</sup>
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

# Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than J
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j => S'\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

# Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j ⇒ S\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

## Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j ⇒ S'\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

# Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j => S'\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

# Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j ⇒ S'\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

# Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j ⇒ S'\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until S = S





Modified Smith Rule with MPs

800000000

## Theorem: Optimality of MSR with MPs

MSR with MPs is optimal for  $1|r_j$ ; *pmtn*;  $MP| \sum C_j$ .

- S = solution of MSR with MPs ; S\* = optimal solution
- *i* has a higher priority than *j*
- Starting at t, in S\*, take out the (non MP) parts of i and j and re-schedule i before j ⇒ S'\*
- Lemma: the objective value of S'\* is no greater than those of S\*
- Repeat until  $S = S^*$



GiSCOP

## Lemma: Optimality of MSR with MP

The objective value of  $S'^*$  is no greater than those of  $S^*$ .

### Notations

- $\widehat{p}_k$  = remaining time of k
- $C_k$  (resp.  $C'_k$ ) = completion time of k in  $S^*$  (resp. in  $S'^*$ )
- $eft_k = earliest finishing time of k (start of the MP)$
- $lst_k = latest starting time of k (end of the MP)$
GiSCOP

### Lemma: Optimality of MSR with MP

The objective value of  $S'^*$  is no greater than those of  $S^*$ .

#### Notations

- $\widehat{p}_k$  = remaining time of k
- $C_k$  (resp.  $C'_k$ ) = completion time of k in  $S^*$  (resp. in  $S'^*$ )
- $eft_k = earliest finishing time of k (start of the MP)$
- $lst_k = latest starting time of k (end of the MP)$



#### Lemma: Optimality of MSR with MP

The objective value of  $S'^*$  is no greater than those of  $S^*$ .

#### Notations

- $\widehat{p}_k$  = remaining time of k
- $C_k$  (resp.  $C'_k$ ) = completion time of k in  $S^*$  (resp. in  $S'^*$ )
- $eft_k$  = earliest finishing time of k (start of the MP)
- $lst_k = latest starting time of k (end of the MP)$



- Case 1: No MP after  $t \Rightarrow$  MSR
- Case 2: MP of i or j in the middle of its execution in  $S'^*$
- Case 3: *i* or *j* ends by its MP in  $S'^*$

#### Goal

- Show that the objective value of S'\* is no greater than those of S\*
- ie. show that  $C_i + C_j \leq C_i + C_j$



• Case 1: No MP after  $t \Rightarrow MSR$ 

• Case 2: MP of *i* or *j* in the middle of its execution in  $S'^*$ 

• Case 5. For J ends by its within a

# Goal • Show that the objective value of S<sup>™</sup> is no greater than those of S<sup>™</sup> • is, show that C[ + C] ≤ C<sub>i</sub> + C<sub>j</sub>



• Case 1: No MP after  $t \Rightarrow MSR$ 

• Case 2: MP of i or j in the middle of its execution in  $S'^*$ 

Case 3: i or j ends by its MP in S<sup>\*</sup>

#### Goal

- Show that the objective value of S<sup>\*</sup> is no greater than those of S<sup>\*</sup>
- ie. show that  $C_i + C_i \leq C_i + C_j$



- Case 1: No MP after  $t \Rightarrow MSR$
- Case 2: MP of i or j in the middle of its execution in  $S'^*$
- Case 3: i or j ends by its MP in S'\*

#### Goal

- Show that the objective value of S'\* is no greater than those of S\*
- ie. show that  $C_i' + C_j \le C_i + C_j$



- Case 1: No MP after  $t \Rightarrow MSR$
- Case 2: MP of i or j in the middle of its execution in  $S'^*$
- Case 3: *i* or *j* ends by its MP in S'\*





- Case 1: No MP after  $t \Rightarrow MSR$
- Case 2: MP of i or j in the middle of its execution in  $S'^*$
- Case 3: *i* or *j* ends by its MP in *S*<sup>\*</sup>

#### Goal

- Show that the objective value of S'\* is no greater than those of S\*
- ie. show that  $C'_i + C'_i \leq C_i + C_j$



- Case 1: No MP after  $t \Rightarrow MSR$
- Case 2: MP of i or j in the middle of its execution in  $S'^*$
- Case 3: *i* or *j* ends by its MP in S'\*

#### Goal

• Show that the objective value of S'\* is no greater than those of S\*

• ie. show that  $C'_i + C'_i \leq C_i + C_j$ 



- Case 1: No MP after  $t \Rightarrow MSR$
- Case 2: MP of i or j in the middle of its execution in  $S'^*$
- Case 3: *i* or *j* ends by its MP in S'\*

#### Goal

- Show that the objective value of S'\* is no greater than those of S\*
- ie. show that  $C'_i + C'_i \leq C_i + C_j$



# Lemma: proof of case 2

### Case 2: MP of i or j in the middle of its execution in $S'^*$

- Definition: MPs are at the same place in  $S^*$  and  $S'^*$
- ullet MPs in the middle  $\Rightarrow$  tasks scheduled around the MPs
- These MPs have no influence on  $C'_i$  and  $C'_i$
- ullet We can ignore these MPs  $\Rightarrow$  go to case 1 or 3



# Lemma: proof of case 2

### Case 2: MP of *i* or *j* in the middle of its execution in $S'^*$

Modified Smith Rule with MPs

- Definition: MPs are at the same place in S\* and S'\*
- ullet MPs in the middle  $\Rightarrow$  tasks scheduled around the MPs
- These MPs have no influence on  $C'_i$  and  $C'_i$
- We can ignore these MPs  $\Rightarrow$  go to case 1 or 3









# Lemma: proof of case 2

Case 2: MP of *i* or *j* in the middle of its execution in  $S'^*$ 

0000000000

Modified Smith Rule with MPs

- Definition: MPs are at the same place in  $S^*$  and  $S'^*$
- MPs in the middle  $\Rightarrow$  tasks scheduled around the MPs
- These MPs have no influence on C<sub>i</sub> and C<sub>i</sub>
- ullet We can ignore these MPs  $\Rightarrow$  go to case 1 or 3





- Definition: MPs are at the same place in  $S^*$  and  $S'^*$
- MPs in the middle  $\Rightarrow$  tasks scheduled around the MPs
- These MPs have no influence on  $C'_i$  and  $C'_i$

ullet We can ignore these MPs  $\Rightarrow$  go to case 1 or 3





# Lemma: proof of case 2

Case 2: MP of *i* or *j* in the middle of its execution in  $S'^*$ 

0000000000

Modified Smith Rule with MPs

- Definition: MPs are at the same place in  $S^*$  and  $S'^*$
- ullet MPs in the middle  $\Rightarrow$  tasks scheduled around the MPs
- These MPs have no influence on  $C'_i$  and  $C'_i$
- We can ignore these MPs  $\Rightarrow$  go to case 1 or 3





Modified Smith Rule with MPs

### Case 3: *i* or *j* ends by its MP in $S'^*$

3 sub-cases:

- Sub-case a): *j* ends with its MP
- Sub-case b): *i* ends with its MP
- Sub-case c): *j* and *i* end with their MP



Modified Smith Rule with MPs

### Case 3: *i* or *j* ends by its MP in $S'^*$

#### 3 sub-cases:

- Sub-case a): *j* ends with its MP
- Sub-case b): *i* ends with its MP
- Sub-case c): *j* and *i* end with their MP



Modified Smith Rule with MPs

### Case 3: *i* or *j* ends by its MP in $S'^*$

#### 3 sub-cases:

- Sub-case a): *j* ends with its MP
- Sub-case b): i ends with its MP
- Sub-case c): j and i end with their MP



Modified Smith Rule with MPs

### Case 3: *i* or *j* ends by its MP in $S'^*$

3 sub-cases:

- Sub-case a): *j* ends with its MP
- Sub-case b): *i* ends with its MP
- Sub-case c): j and i end with their MP



Modified Smith Rule with MPs

### Case 3: *i* or *j* ends by its MP in $S'^*$

3 sub-cases:

- Sub-case a): *j* ends with its MP
- Sub-case b): *i* ends with its MP
- Sub-case c): *j* and *i* end with their MP

G-SCOP

Modified Smith Rule with MPs

Experimentatio

Conclusion 0000

# Lemma: proof of case 3 a)

### Sub-case a): *j* ends with its MP

- $C_j = C'_i = eft_j$
- Construction: i processed at t in  $S'^*$  (1)
- Construction: j processed in  $[t, eft_j]$  in  $S'^*$  (2)
- $(1) + (2) \Rightarrow \text{CONTRADICTION} !$



## Sub-case a): *j* ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in  $S'^*$  (1)
- Construction: j processed in  $[t, eft_j]$  in  $S'^*$  (2)
- $(1) + (2) \Rightarrow \text{CONTRADICTION}$  !





### Sub-case a): *j* ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in  $S'^*$  (1)
- Construction: j processed in  $[t, eft_j]$  in  $S'^*$  (2)
- $(1) + (2) \Rightarrow \text{CONTRADICTION} !$





Modified Smith Rule with MPs

Experimentation

Conclusion

# Lemma: proof of case 3 a)

### Sub-case a): j ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in  $S'^*$  (1)
- Construction: j processed in  $[t, eft_j[$  in  $S'^*(2)$

•  $(1) + (2) \Rightarrow \text{CONTRADICTION} !$ 





Modified Smith Rule with MPs

Experimentation

Conclusion 0000

# Lemma: proof of case 3 a)

### Sub-case a): *j* ends with its MP

- $C_j = C'_j = eft_j$
- Construction: i processed at t in  $S'^*$  (1)
- Construction: j processed in  $[t, eft_j]$  in  $S'^*$  (2)
- $(1) + (2) \Rightarrow \text{CONTRADICTION} !$





### Sub-case c): *i* and *j* ends with their MP

- Sub-case a): j ends with its MP IMPOSSIBLE !
- Sub-case c): *i* and *j* ends with their MP  $\Rightarrow$  IMPOSSIBLE



#### Sub-case c): *i* and *j* ends with their MP

- Sub-case a): *j* ends with its MP IMPOSSIBLE !
- Sub-case c): *i* and *j* ends with their MP  $\Rightarrow$  IMPOSSIBLE

000000000

Modified Smith Rule with MPs





#### Sub-case c): *i* and *j* ends with their MP

- Sub-case a): *j* ends with its MP IMPOSSIBLE !
- Sub-case c): *i* and *j* ends with their MP  $\Rightarrow$  IMPOSSIBLE !

0000000000

Modified Smith Rule with MPs



G-SCOP

Introduction

Modified Smith Rule with MPs

Experimentatio

Conclusion 0000

# Lemma: proof of case 3 b)

### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - G = G



## Modified Smith Rule with MPs 00000000 Lemma: proof of case 3 b) Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$





Modified Smith Rule with MPs

00000000

### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$





Modified Smith Rule with MPs

00000000

### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time  $t : C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$

•  $\vec{C}'_i = C$ 





Modified Smith Rule with MPs

00000000

### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_j = C_j$  and  $(1) \Rightarrow C'_i + C'_j < C_i + C_j$





Modified Smith Rule with MPs

00000000

### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time  $t : C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_j = C_j$  and (1)  $\Rightarrow C'_i + C'_j < C_i + C_j$





Modified Smith Rule with MPs

00000000

### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time  $t : C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_j = C_j$  and  $(1) \Rightarrow C'_i + C'_j < C_i + C_j$




#### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_i = C_j$  and  $(1) \Rightarrow C'_i + C'_i < C_i + C_j$
  - $\widetilde{C'_j} = C_i$ :  $\widehat{p}_i < \widehat{p}_j$  (rule)  $\Rightarrow$   $eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

Modified Smith Rule with MPs





#### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_j = C_j$  and  $(1) \Rightarrow C'_i + C'_j < C_i + C_j$
  - $\widetilde{C}'_j = C_i : \widehat{p}_i < \widehat{p}_j \text{ (rule)} \Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

Modified Smith Rule with MPs

00000000





#### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_j = C_j$  and  $(1) \Rightarrow C'_i + C'_j < C_i + C_j$
  - $\check{C}'_j = C_i : \widehat{p}_i < \widehat{p}_j \text{ (rule)} \Rightarrow eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

Modified Smith Rule with MPs

00000000





#### Sub-case b): *i* ends with its MP

- Construction: *i* processed in  $[t, eft_i]$  in  $S'^*$
- $S^* \neq S'^*$  at time t :  $C_i > C'_i = eft_i$  (1)
- $C'_j = \max(C_j, C_i)$ 
  - $C'_j = C_j$  and  $(1) \Rightarrow C'_i + C'_j < C_i + C_j$
  - $\widetilde{C'_j} = \widetilde{C_i}$  :  $\widehat{p_i} < \widehat{p_j}$  (rule)  $\Rightarrow$   $eft_i < C_j \Rightarrow C'_i + C'_j < C_i + C_j$

Modified Smith Rule with MPs

00000000



roduction Modified Smith Rule with MPs Experimentations Conc OO Occorrection



# 3. Experimentations



ntroduction Mod

Modified Smith Rule with M

# **Experimental protocol**

#### 570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- $\bullet$  with durations in [1, 10] or [1, 100]
- with time windows [r<sub>j</sub>, d<sub>j</sub>] more or less close of those of a **possible schedule**



troduction Modified Sn

# Experimental protocol

#### 570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- $\bullet$  with durations in [1,10] or [1,100]
- with time windows [r<sub>j</sub>, d<sub>j</sub>] more or less close of those of a **possible schedule**



roduction Modified Smi

dified Smith Rule with MP 0000000

# **Experimental protocol**

#### 570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- $\bullet$  with durations in [1,10] or [1,100]
- with time windows [r<sub>j</sub>, d<sub>j</sub>] more or less close of those of a **possible schedule**

#### Problems studied



tion Modified Smith Rule with

# **Experimental protocol**

#### 570 instances generated ...

- with 20, 40, 60, 80 or 100 tasks
- $\bullet$  with durations in [1,10] or [1,100]
- with time windows [r<sub>j</sub>, d<sub>j</sub>] more or less close of those of a **possible schedule**

# Problems studiedRD-CP $|i_1; d_j| \sum G$ OPTRDP-MP-IP $|i_1; d_j; pmtn; MP| \sum G$ LBRDP-IP $|i_1; d_j; pmtn; MP| \sum G$ LBRP-MP-S $|i_1; pmtn; MP| \sum G$ LBRP-S $|i_1; pmtn| \sum G$ LB



tion Modified Smith Rule wit

# **Experimental protocol**

#### 570 instances generated . . .

- with 20, 40, 60, 80 or 100 tasks
- $\bullet$  with durations in [1,10] or [1,100]
- with time windows [r<sub>j</sub>, d<sub>j</sub>] more or less close of those of a **possible schedule**

#### Problems studied

RP-MP-S	



17/21

#### n Modified Smith Rule with MPs

# **Experimental protocol**

#### 570 instances generated . . .

- with 20, 40, 60, 80 or 100 tasks
- $\bullet$  with durations in [1,10] or [1,100]
- with time windows [r<sub>j</sub>, d<sub>j</sub>] more or less close of those of a **possible schedule**

Problem	is studied			
RD-CP RDP-MP-IP RDP-IP	$\begin{array}{l} 1 r_j; d_j  \sum C_j \\ 1 r_j; d_j; pmtn; MP  \sum C_j \\ 1 r_j; d_j; pmtn  \sum C_j \end{array}$	OPT LB LB		
	RP-MP-S RP-S	$1 r_j; pmtn; MP  \sum C_j$ $1 r_j; pmtn  \sum C_j$	LB LB	



Modified Smith Rule with

Experimentations

Conclusion 0000

# Experimentations

## Gaps to the optimum analysis: $LB \div OPT$ (RD-CP)

#### Remarks

- RP-S  $\leq$  RP-MP-S  $\leq$  RDP-IP  $\leq$  RDP-MP-IP
- Less results (nb gaps) for IP-solved problems (NP-hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is



Modified Smith Rule witl 0000000000 Experimentations

Conclusion 0000

# Experimentations

# Gaps to the optimum analysis: $LB \div OPT (RD-CP)$

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

#### Remarks

- RP-S  $\leq$  RP-MP-S  $\leq$  RDP-IP  $\leq$  RDP-MP-IP
- Less results (nb gaps) for IP-solved problems ( $\mathcal{NP}$ -hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is



Modified Smith Rule with

Experimentations

Conclusion 0000

# Experimentations

## Gaps to the optimum analysis: $LB \div OPT (RD-CP)$

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

#### Remarks

- $\mathsf{RP-S} \leq \mathsf{RP-MP-S} \leq \mathsf{RDP-IP} \leq \mathsf{RDP-MP-IP}$
- Less results (nb gaps) for IP-solved problems ( $\mathcal{NP}$ -hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is



Modified Smith Rule with

Experimentations

Conclusion 0000

# Experimentations

# Gaps to the optimum analysis: $LB \div OPT (RD-CP)$

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

#### Remarks

- $\mathsf{RP}$ - $\mathsf{S} \leq \mathsf{RP}$ - $\mathsf{MP}$ - $\mathsf{S} \leq \mathsf{RDP}$ - $\mathsf{IP} \leq \mathsf{RDP}$ - $\mathsf{MP}$ - $\mathsf{IP}$
- Less results (nb gaps) for IP-solved problems ( $\mathcal{NP}$ -hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is



Modified Smith Rule with

Experimentations

Conclusion 0000

# Experimentations

## Gaps to the optimum analysis: $LB \div OPT (RD-CP)$

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

#### Remarks

- $RP-S \leq RP-MP-S \leq RDP-IP \leq RDP-MP-IP$
- Less results (nb gaps) for IP-solved problems ( $\mathcal{NP}$ -hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is



Modified Smith Rule w S SOOOOOOOOO Experimentations

Conclusion 0000

# Experimentations

### Gaps to the optimum analysis: $LB \div OPT (RD-CP)$

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

#### Remarks

- $RP-S \leq RP-MP-S \leq RDP-IP \leq RDP-MP-IP$
- Less results (nb gaps) for IP-solved problems ( $\mathcal{NP}$ -hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is

• For all problems : the closer the instance is to a solution, the tinier the gap to OPT is



Modified Smith Rule with 0000000000 Experimentations

Conclusion

# Experimentations

# Gaps to the optimum analysis: $LB \div OPT$ (RD-CP)

Problem	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Gaps
RDP-MP-IP	0.9830	0.9994	0.9999	0.9993	1.0000	1.0000	258
RDP-IP	0.9830	0.9991	0.9998	0.9990	1.0000	1.0000	239
RP-MP-S	0.9575	0.9978	0.9995	0.9980	1.0000	1.0000	403
RP-S	0.9566	0.9967	0.9990	0.9973	0.9998	1.0000	403

#### Remarks

- $\mathsf{RP}$ - $\mathsf{S} \leq \mathsf{RP}$ - $\mathsf{MP}$ - $\mathsf{S} \leq \mathsf{RDP}$ - $\mathsf{IP} \leq \mathsf{RDP}$ - $\mathsf{MP}$ - $\mathsf{IP}$
- Less results (nb gaps) for IP-solved problems ( $\mathcal{NP}$ -hard)
- For all problems : the smaller the nb of tasks, the tinier the gap to OPT is
- For all problems : the closer the instance is to a solution, the tinier the gap to OPT is

oduction Modified Smith Rule with MPs Experimentations Conclusion

# G-SCOP

# 4. Conclusion

# Conclusion

#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of  $d_j$

# Conclusion

#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of  $d_j$



#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j$ ;  $d_j$ ;  $pmtn|\sum C_j$
- No direct integration of  $d_j$

Modified Smith Rule with MP 000000000 Conclusion ○●○○



#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of  $d_i$



#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of  $d_i$

Conclusion



#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of  $d_i$

Conclusion



#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of  $d_j$

Conclusion



#### Conclusion

- MSR with MP compute LB for  $1|r_j; d_j| \sum C_j$  in poly time
- Improve the LB compared to MSR
- Does not complexify too much MSR
- Can be used for improving CP resolution
- MP can be constrained for practical applications : personnel needed for a 2-persons tasks in an hospital
- Possible improving toward value of  $1|r_j; d_j; pmtn| \sum C_j$
- No direct integration of d<sub>i</sub>



lodified Smith Rule with MP

Experimentation: 000 Conclusion



#### Prospects

- Partial integration of d<sub>i</sub>
- CP applications
- • •



on Modified Smith Rule with

Experimentation

Conclusion



#### Prospects

- Partial integration of  $d_i$
- CP applications

••••



n Modified Smith Rule with MPs Ex

Experimentations

Conclusion



#### Prospects

- Partial integration of  $d_j$
- CP applications

• • • •





#### Prospects

- Partial integration of  $d_i$
- CP applications
- • •

uction Modified Smith Rule with MPs Experiment



# Thank you for your attention !



# 5. Appendix

Appendix

•0

G. SCOP

# Sweep line algo VS list algo

#### Differences

• List: sorts elements with the same level

Appendix

- Sweep line: sorts elements with the different levels
- List: all elements have same actions
- Sweep line: elements can have different actions