La contrainte MinCumulative

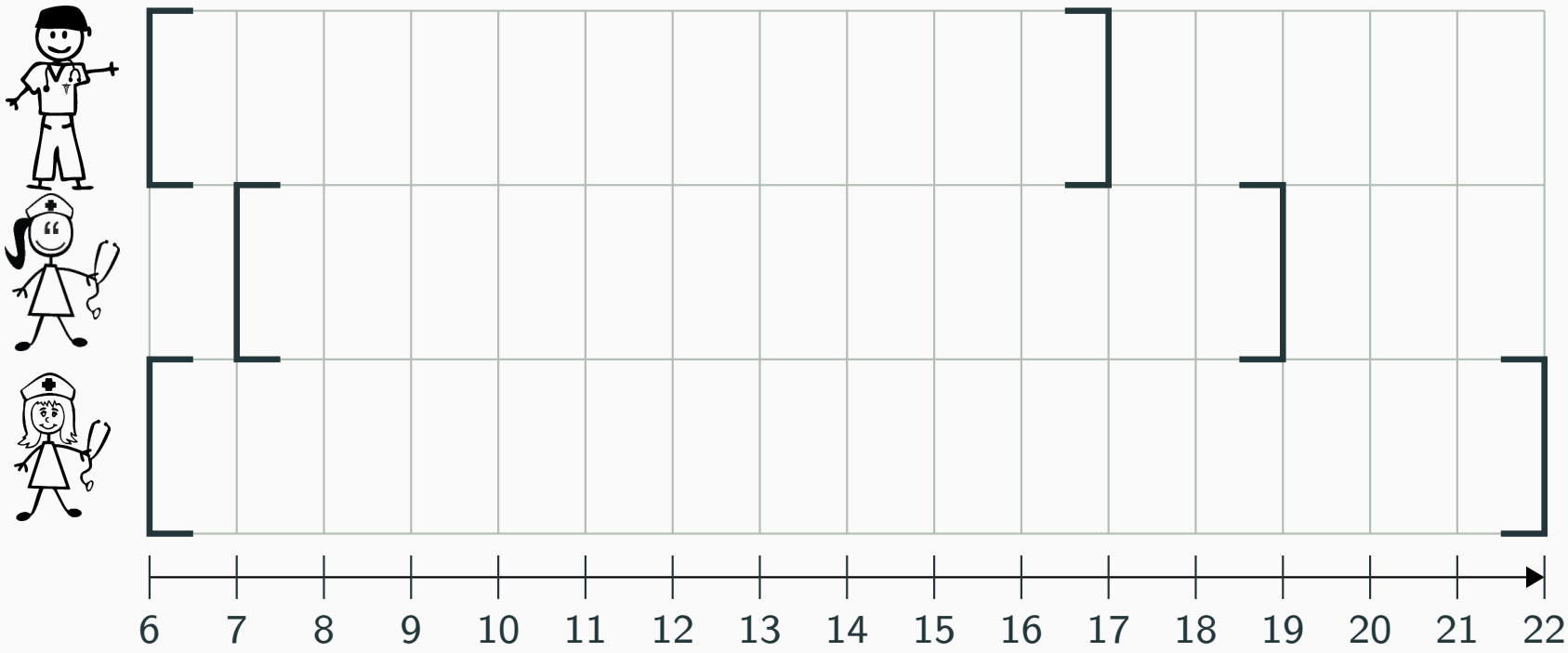
Yanick Ouellet

Claude-Guy Quimper

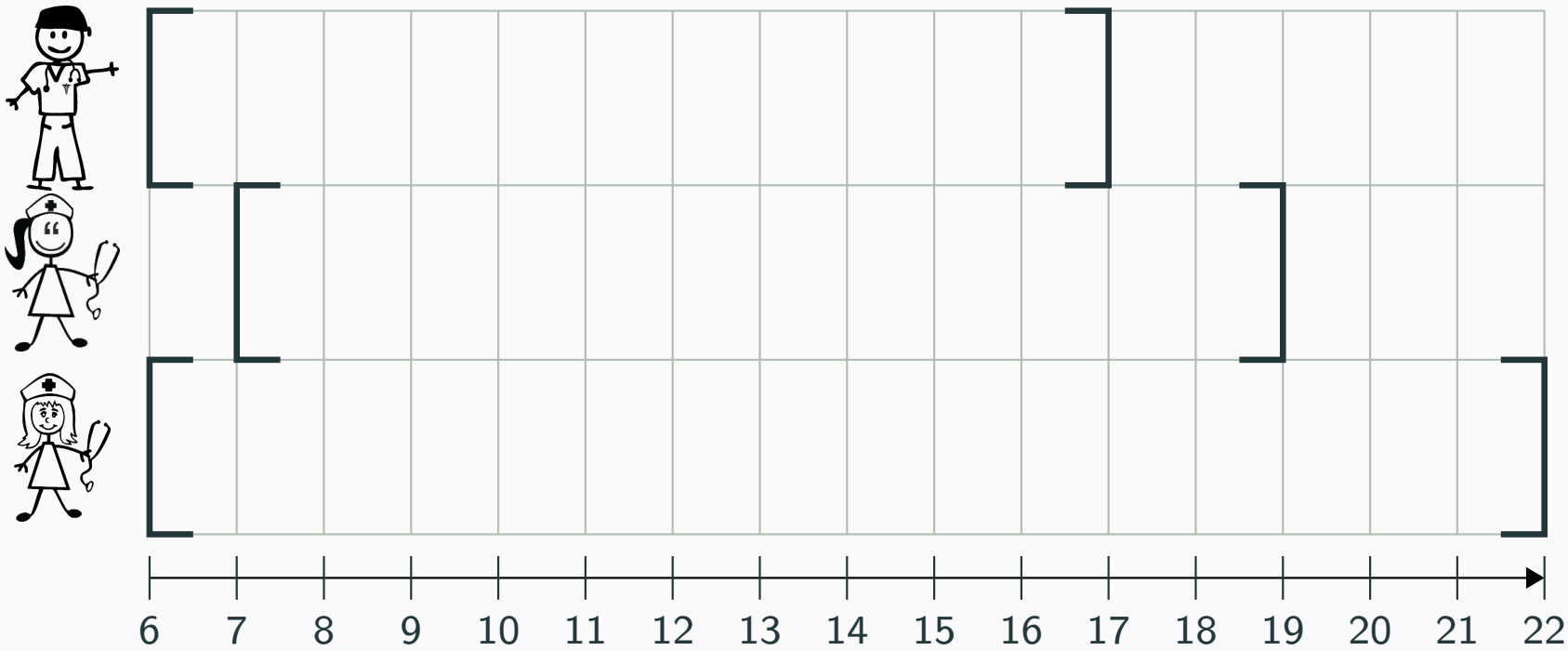
9 avril 2021

Université Laval

Motivation

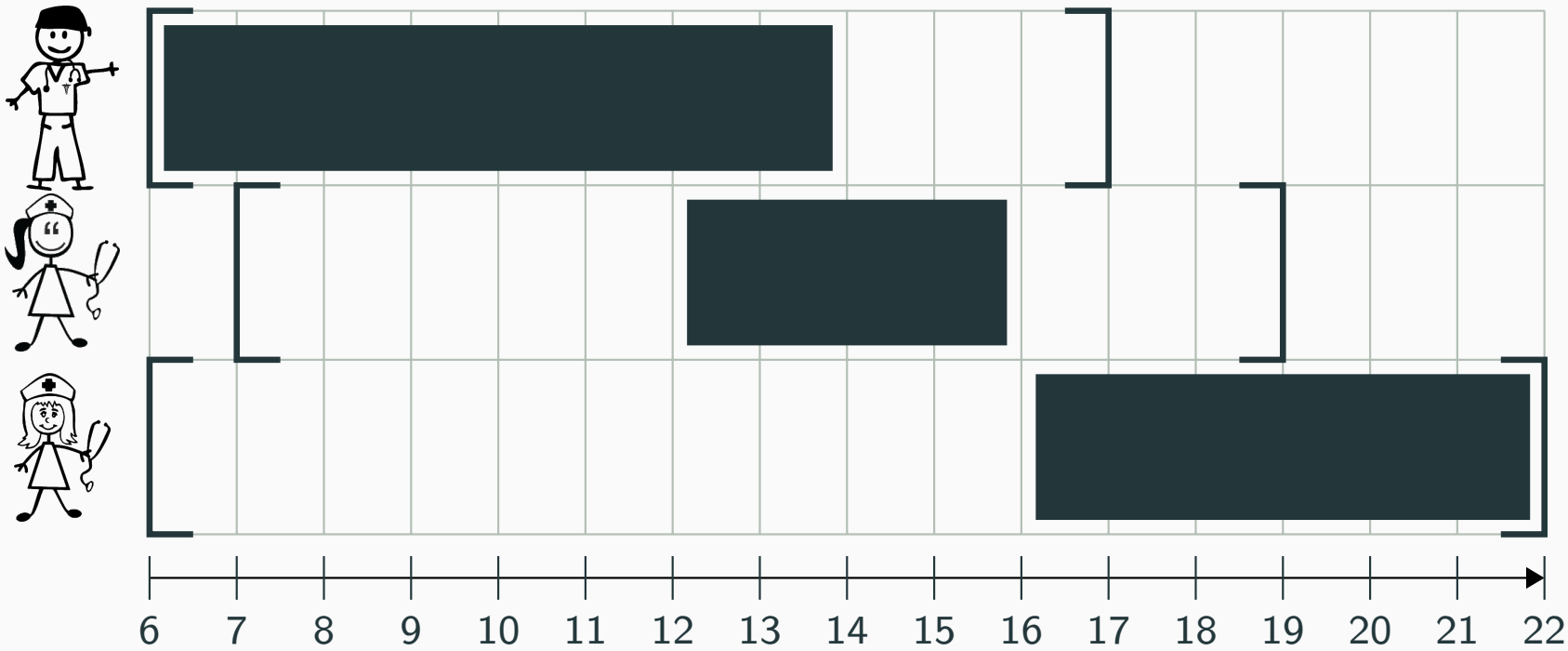


Motivation



Demande	1	1	1	1	1	1	**2**	1	1	1	1	1	1	1	1	1

Motivation



Demande	1	1	1	1	1	1	**2**	1	1	1	1	1	1	1	1	1

Approches pour résoudre le problème

MIP

$$S_1 + p_1 \leq S_2$$

Approches pour résoudre le problème

MIP

$S_1 + p_1 \leq S_2$

PPC

$\texttt{MinCumul}(S, p, h, d)$

Approches pour résoudre le problème

MIP

$S_1 + p_1 \leq S_2$

PPC

$\texttt{MinCumul}(S, p, h, d)$

SAT

$[\![S_1 = 2]\!] \vee [\![S_2 = 2]\!]$

Approches pour résoudre le problème

MIP

$S_1 + p_1 \leq S_2$

PPC

$\texttt{MinCumul}(S, p, h, d)$

SAT

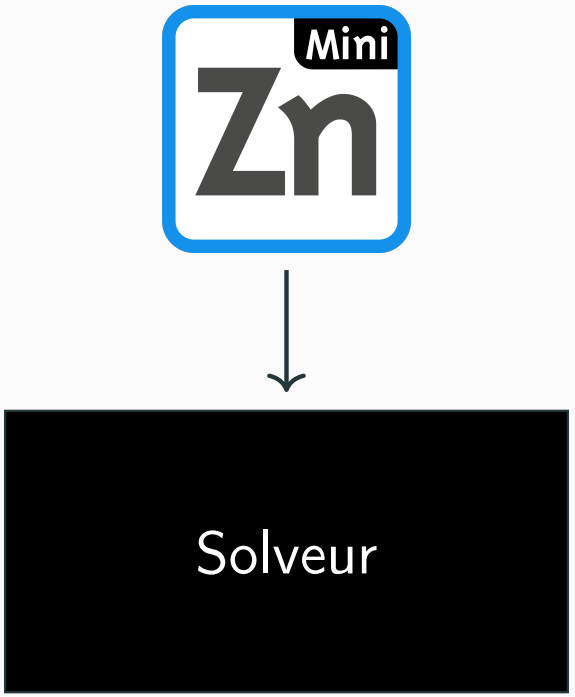
$[\![S_1 = 2]\!] \vee [\![S_2 = 2]\!]$

Méta-heuristiques

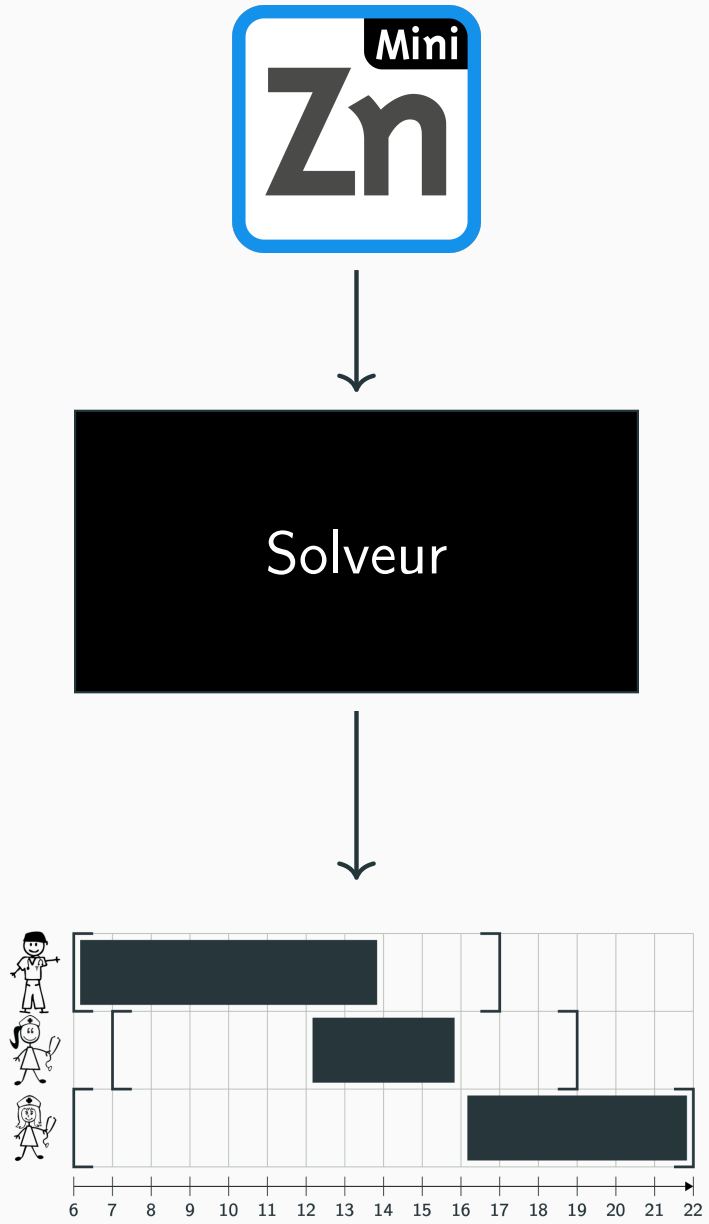
$\texttt{def lns}(S, p, h, d)\texttt{:}$

Programmation par contraintes

Programmation par contraintes



Programmation par contraintes



Exemple d'un modèle

Problème

Assigner un quart de travail à **R**obert, **N**ancy et **M**onique.

Exemple d'un modèle

Problème

Assigner un quart de travail à **R**obert, **N**ancy et **M**onique.

Variables

- $R \in \{\texttt{AM}, \texttt{PM}\}$
- $N \in \{\texttt{AM}, \texttt{PM}\}$
- $M \in \{\texttt{AM}, \texttt{PM}\}$

Exemple d'un modèle

Problème

Assigner un quart de travail à **R**obert, **N**ancy et **M**onique.

Variables

- $R \in \{\texttt{AM}, \texttt{PM}\}$
- $N \in \{\texttt{AM}, \texttt{PM}\}$
- $M \in \{\texttt{AM}, \texttt{PM}\}$

Contraintes

- Au moins deux quarts en `AM`
- Au moins un quart en `PM`
- Robert ne peut pas travailler avec Monique ($R \neq M$)

Fonctionnement d'un solveur

Principe

- Construit un arbre de recherche

Fonctionnement d'un solveur

Principe

- Construit un arbre de recherche

Heuristique de branchement

- Choisis une variable et une valeur à assigner

Fonctionnement d'un solveur

Principe

- Construit un arbre de recherche

Heuristique de branchement

- Choisis une variable et une valeur à assigner

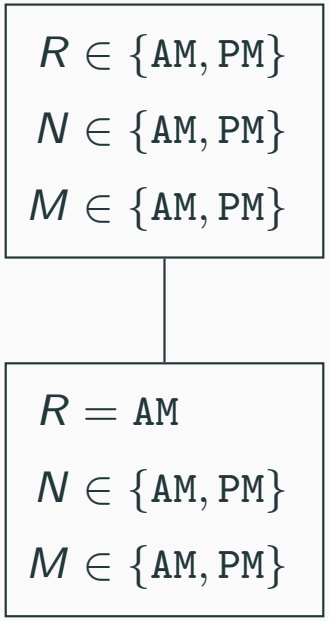
Contraintes

- Algorithme de vérification
- Algorithme de filtrage

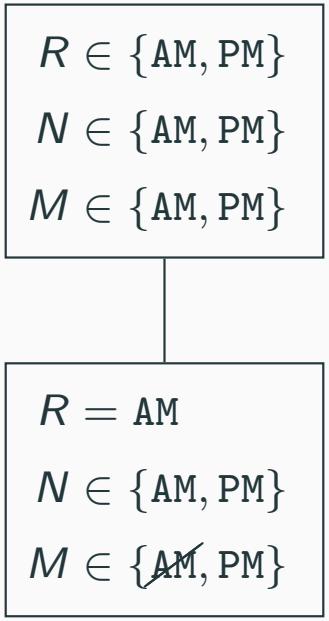
Arbre de recherche

$$R \in \{\texttt{AM}, \texttt{PM}\}$$
$$N \in \{\texttt{AM}, \texttt{PM}\}$$
$$M \in \{\texttt{AM}, \texttt{PM}\}$$

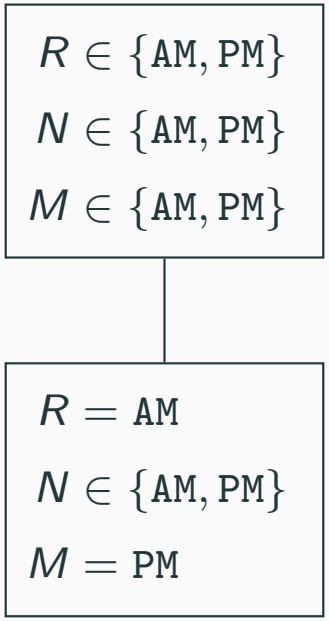
Arbre de recherche



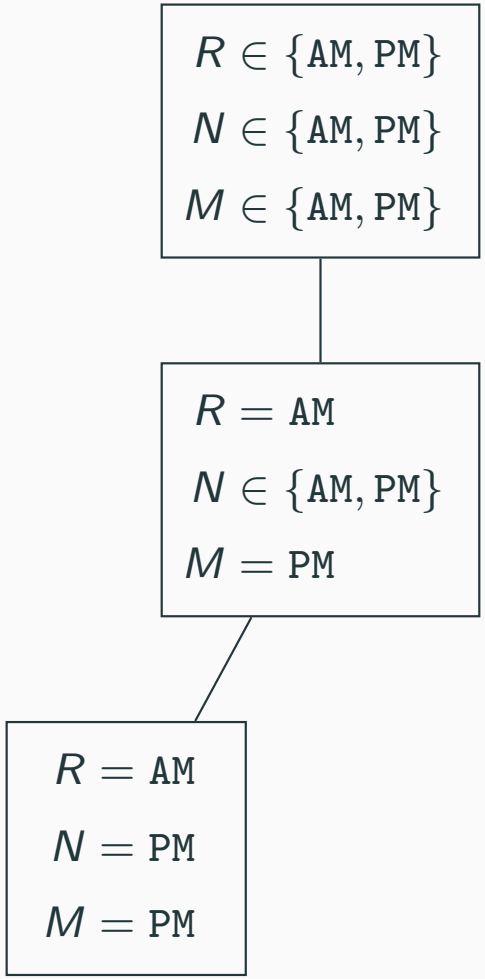
Arbre de recherche



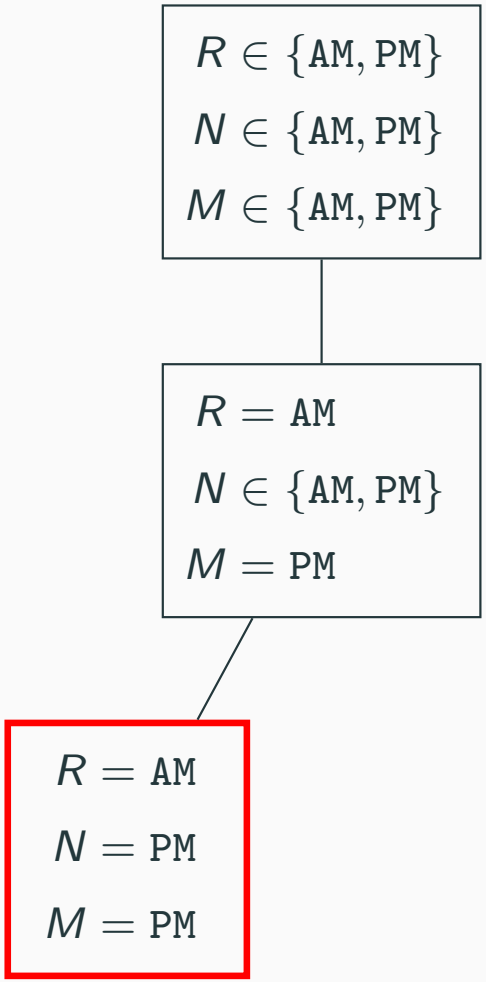
Arbre de recherche



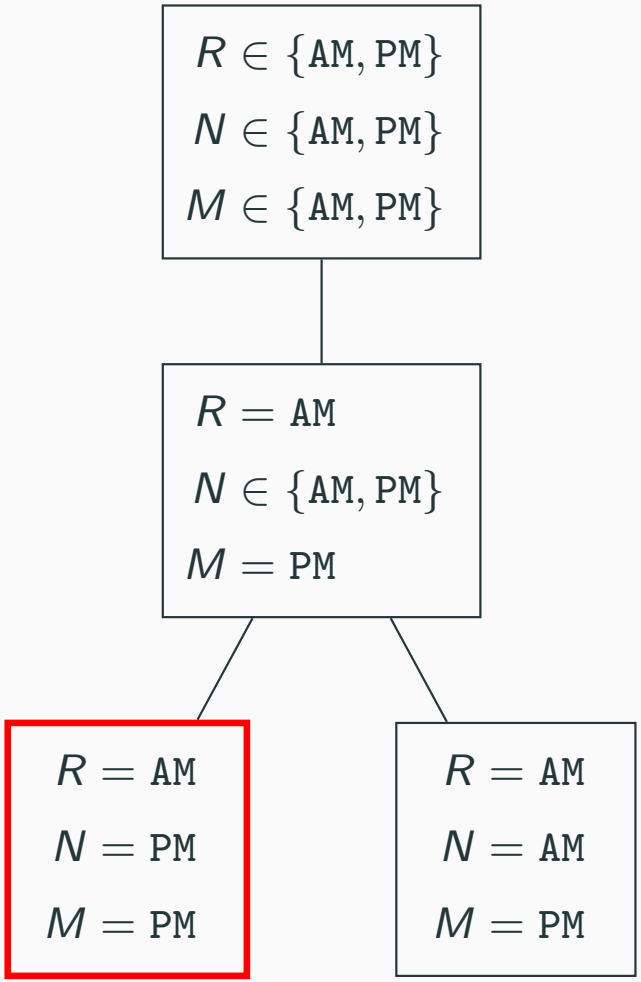
Arbre de recherche



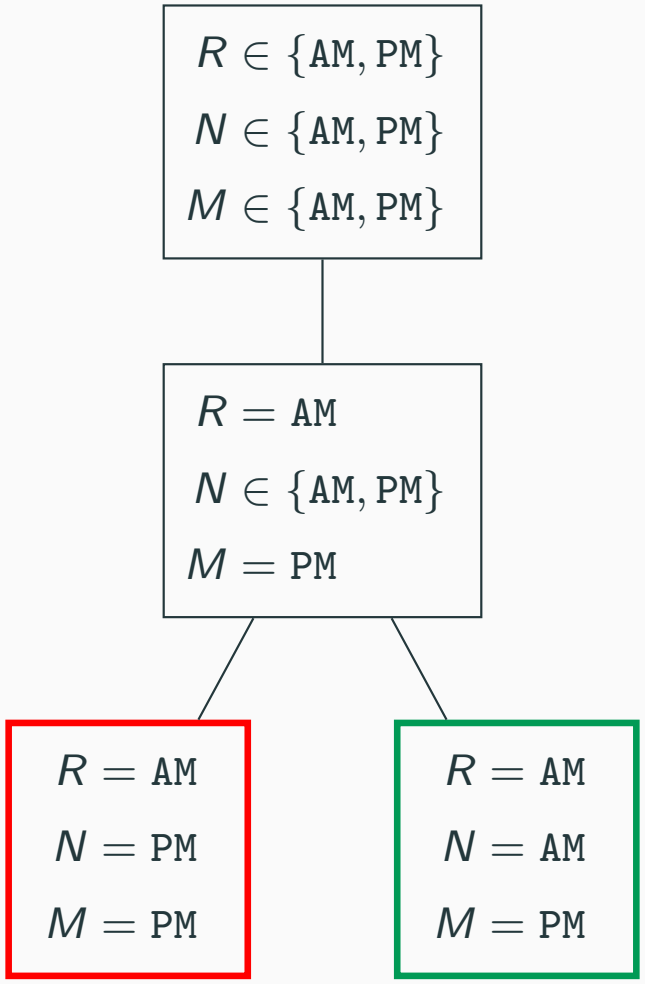
Arbre de recherche



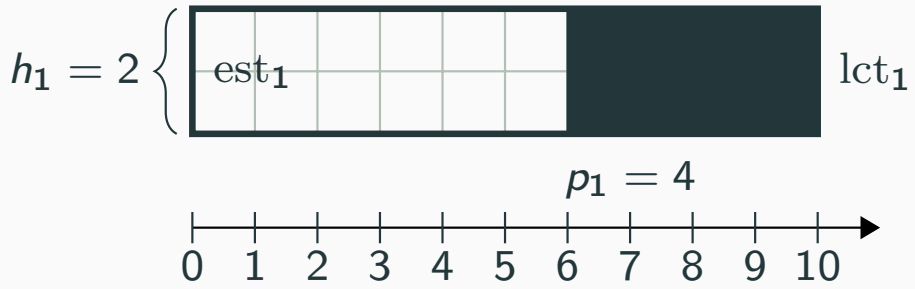
Arbre de recherche



Arbre de recherche



Définition d'une tâche



Notation

- est : Earliest Starting Time
- lct : Lastest Completion Time
- p : Processing time
- h : Height

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Variables et paramètres

- $\mathcal{I}$: Ensemble des tâches

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Variables et paramètres

- $\mathcal{I}$: Ensemble des tâches
- T : Ensemble des instants

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Variables et paramètres

- $\mathcal{I}$: Ensemble des tâches
- T : Ensemble des instants
- $S_i \in [\mathrm{est}_i, \mathrm{lct}_i - p_i]$: Dates de début de $i \in \mathcal{I}$

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Variables et paramètres

- $\mathcal{I}$: Ensemble des tâches
- T : Ensemble des instants
- $S_i \in [\mathrm{est}_i, \mathrm{lct}_i - p_i]$: Dates de début de $i \in \mathcal{I}$
- p_i : Temps de traitement de $i \in \mathcal{I}$

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Variables et paramètres

- $\mathcal{I}$: Ensemble des tâches
- T : Ensemble des instants
- $S_i \in [\mathrm{est}_i, \mathrm{lct}_i - p_i]$: Dates de début de $i \in \mathcal{I}$
- p_i : Temps de traitement de $i \in \mathcal{I}$
- h_i : Hauteur de $i \in \mathcal{I}$

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Variables et paramètres

- $\mathcal{I}$: Ensemble des tâches
- T : Ensemble des instants
- $S_i \in [\mathrm{est}_i, \mathrm{lct}_i - p_i]$: Dates de début de $i \in \mathcal{I}$
- p_i : Temps de traitement de $i \in \mathcal{I}$
- h_i : Hauteur de $i \in \mathcal{I}$
- d_t : Demande à l'instant $t \in T$

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Décomposition

$$\sum_{i \in \mathcal{I} | S_i \leq t < S_i + p_i} h_i \geq d_t \ \forall t \in T$$

Contrainte MinCumulative

Définition

$\texttt{MinCumulative}(S, p, h, d)$

Décomposition

$$\sum_{i \in \mathcal{I} | S_i \leq t < S_i + p_i} h_i \geq d_t \ \forall t \in T$$

Objectif

Développer un algorithme de vérification et de filtrage

Nature de la contrainte

NP-Complet

Décider si la contrainte admet une solution est NP-Complet.

Nature de la contrainte

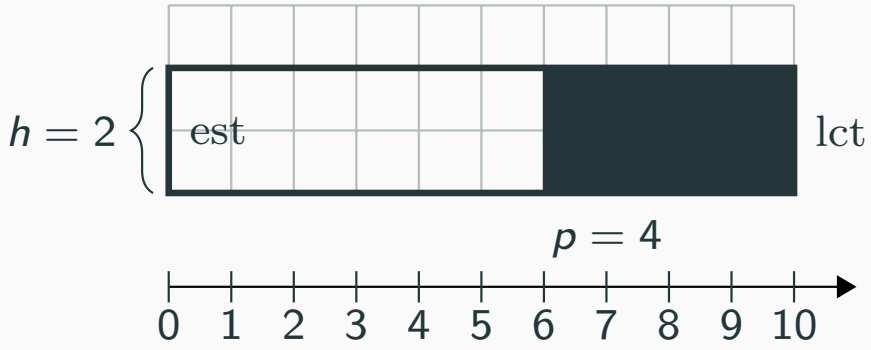
NP-Complet

Décider si la contrainte admet une solution est NP-Complet.

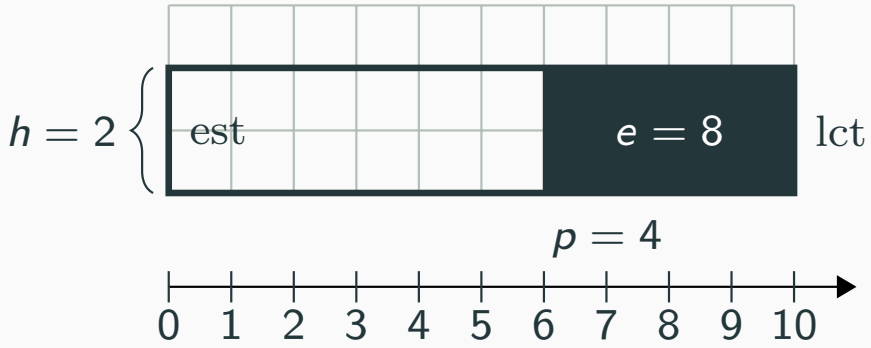
Conséquence

Il faut utiliser une relaxation pour détecter l'incohérence et faire du filtrage.

Relaxation complètement élastique



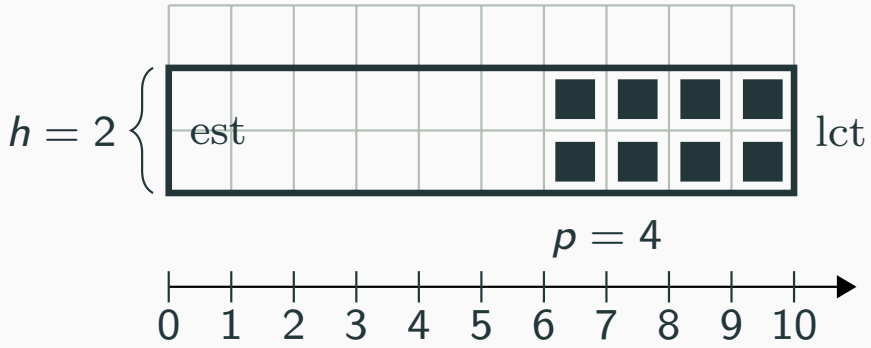
Relaxation complètement élastique



Relaxation

- Énergie : Aire de la tâche $(p \cdot h)$

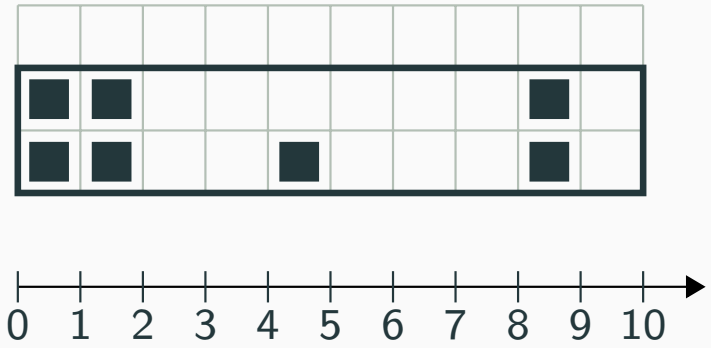
Relaxation complètement élastique



Relaxation

- Énergie : Aire de la tâche $(p \cdot h)$

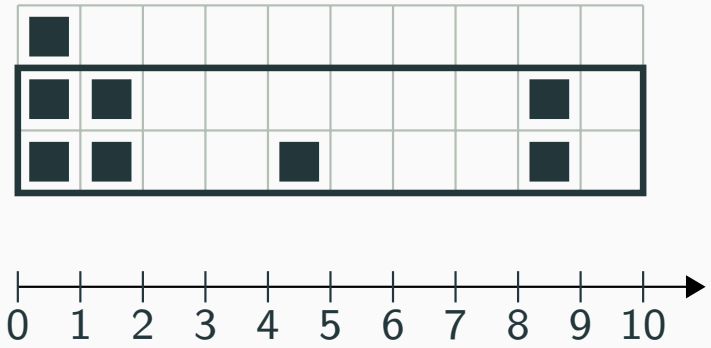
Relaxation complètement élastique



Relaxation

- Énergie : Aire de la tâche ($p \cdot h$)
- On peut placer l'énergie n'importe où dans $[\mathrm{est}, \mathrm{lct})$
  - Préemption possible

Relaxation complètement élastique



Relaxation

- Énergie : Aire de la tâche ($p \cdot h$)
- On peut placer l'énergie n'importe où dans $[\mathrm{est}, \mathrm{lct})$
  - Préemption possible
  - Possible de dépasser la hauteur

Algorithme de vérification

Intuition

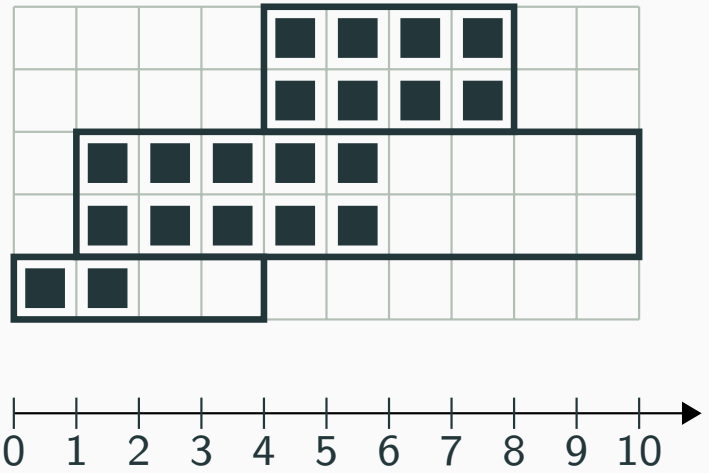
- Utiliser la relaxation complètement élastique
- Trouver une façon de couvrir la demande avec l'énergie

Algorithme de vérification

Intuition

- Utiliser la relaxation complètement élastique
- Trouver une façon de couvrir la demande avec l'énergie

Demande : 2 $\forall t \in [0, 10)$

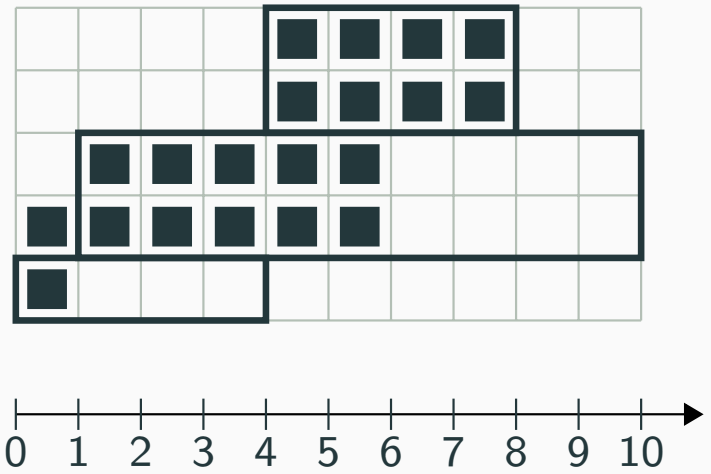


Algorithme de vérification

Intuition

- Utiliser la relaxation complètement élastique
- Trouver une façon de couvrir la demande avec l'énergie

Demande : $2\ \forall t \in [0, 10)$

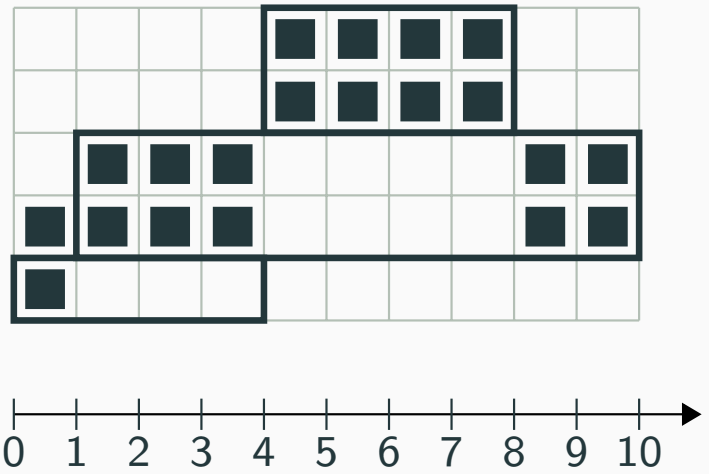


Algorithme de vérification

Intuition

- Utiliser la relaxation complètement élastique
- Trouver une façon de couvrir la demande avec l'énergie

Demande : 2 $\forall t \in [0, 10)$



Algorithme de vérification

Algorithme

1. Trier les tâches par lct

Algorithme de vérification

Algorithme

1. Trier les tâches par lct
2. Traiter chaque tâche i

Algorithme de vérification

Algorithme

1. Trier les tâches par lct
2. Traiter chaque tâche i
    - Céduler son énergie le plus tôt possible

Algorithme de vérification

Algorithme

1. Trier les tâches par lct
2. Traiter chaque tâche i
   - Céduler son énergie le plus tôt possible
   - Sans dépasser la demande ni son lct_i

Algorithme de vérification

Algorithme

1. Trier les tâches par lct
2. Traiter chaque tâche i
   - Céduler son énergie le plus tôt possible
   - Sans dépasser la demande ni son lct_i
3. Si la demande n'est pas entièrement satisfaite, retourner un échec

Algorithme de vérification

Algorithme

1. Trier les tâches par lct
2. Traiter chaque tâche i
   - Céduler son énergie le plus tôt possible
   - Sans dépasser la demande ni son lct_i
3. Si la demande n'est pas entièrement satisfaite, retourner un échec

Complexité

- Linéaire (plus le tri)

Algorithme de vérification

Algorithme

1. Trier les tâches par lct
2. Traiter chaque tâche i
   - Céduler son énergie le plus tôt possible
   - Sans dépasser la demande ni son lct_i
3. Si la demande n'est pas entièrement satisfaite, retourner un échec

Complexité

- Linéaire (plus le tri)
  - Grâce aux ensembles disjoints

Trace

Tâches

#	est	lct	e
1	0	4	2
2	1	10	10
3	4	8	8

Trace

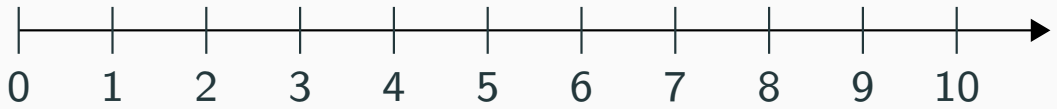
Tâches

#	est	lct	e
1	0	4	2
3	4	8	8
2	1	10	10

Trace

Tâches

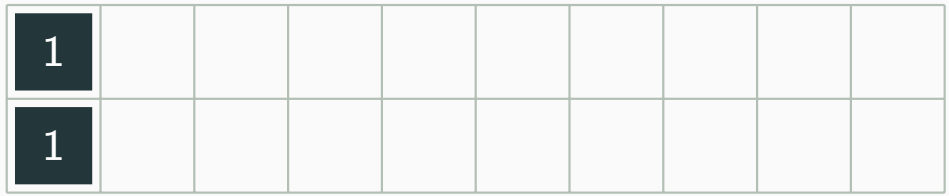
#	est	lct	e
1	0	4	2
3	4	8	8
2	1	10	10

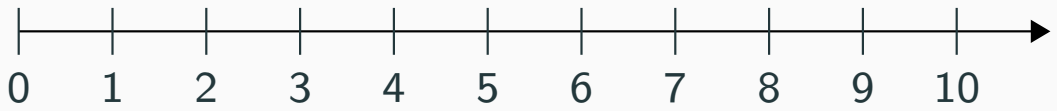


Trace

Tâches

#	est	lct	e
1	**0**	**4**	**2**
3	4	8	8
2	1	10	10

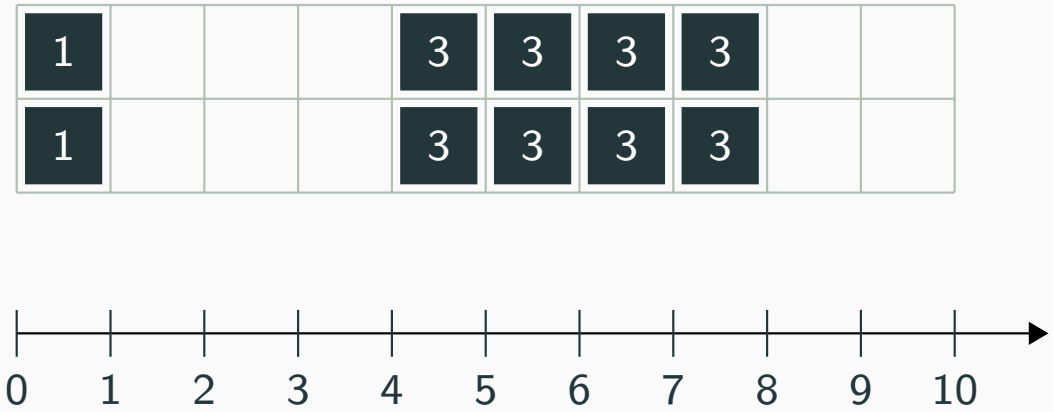




Trace

Tâches

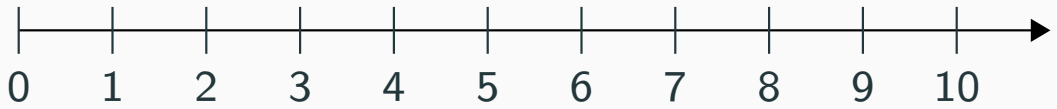
#	est	lct	e
1	0	4	2
3	**4**	**8**	**8**
2	1	10	10



Trace

Tâches

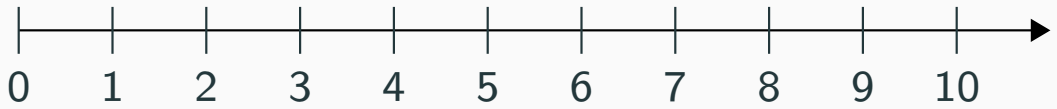
#	est	lct	e
1	0	4	2
3	4	8	8
2	**1**	**10**	**10**



Trace

Tâches

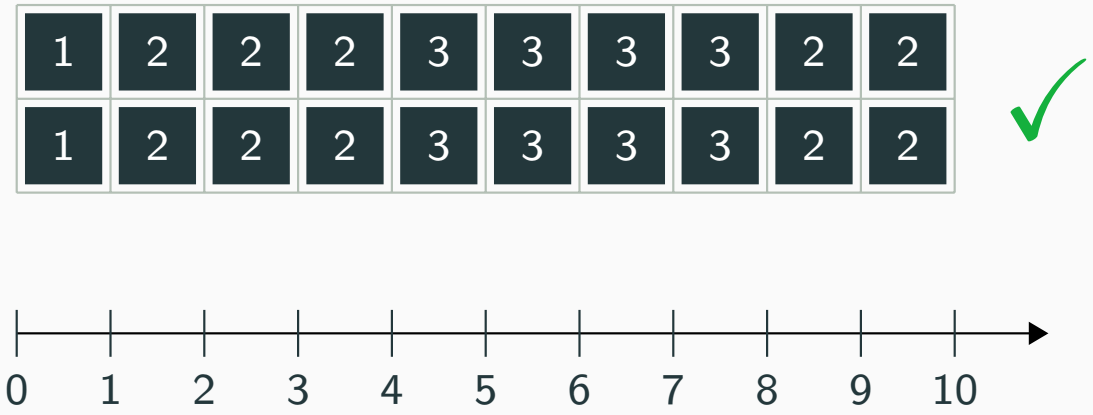
#	est	lct	e
1	0	4	2
3	4	8	8
2	**1**	**10**	**10**



Trace

Tâches

#	est	lct	e
1	0	4	2
3	4	8	8
2	**1**	**10**	**10**



Algorithme de filtrage

Intuition

- Pour chaque tâche i

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i
  4. Filtrer S_i à $\mathrm{est}_i + 1$

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i
  4. Filtrer S_i à $\mathrm{est}_i + 1$
  5. Recommencer jusqu'à ce que l'algorithme de vérification retourne un succès

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i
  4. Filtrer S_i à $\mathrm{est}_i + 1$
  5. Recommencer jusqu'à ce que l'algorithme de vérification retourne un succès

Vérification par la bande

- Vider le domaine d'une tâche $\Rightarrow$ échec

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i
  4. Filtrer S_i à $\mathrm{est}_i + 1$
  5. Recommencer jusqu'à ce que l'algorithme de vérification retourne un succès

Vérification par la bande

- Vider le domaine d'une tâche $\Rightarrow$ échec
- Plus fort que l'algorithme de vérification seul

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i
  4. Filtrer S_i à $\mathrm{est}_i + 1$
  5. Recommencer jusqu'à ce que l'algorithme de vérification retourne un succès

Vérification par la bande

- Vider le domaine d'une tâche $\Rightarrow$ échec
- Plus fort que l'algorithme de vérification seul
  - Pas préemption pour i

Algorithme de filtrage

Intuition

- Pour chaque tâche i
  1. Fixer la tâche à son est_i
  2. Exécuter l'algorithme de vérification
  3. Si l'algorithme échoue, i ne peut pas être cédulée à est_i
  4. Filtrer S_i à $\mathrm{est}_i + 1$
  5. Recommencer jusqu'à ce que l'algorithme de vérification retourne un succès

Vérification par la bande

- Vider le domaine d'une tâche $\Rightarrow$ échec
- Plus fort que l'algorithme de vérification seul
  - Pas préemption pour i
  - Peut quand même dépasser sa hauteur

Trace

Tâches

#	est	lct	e
1	0	4	2
2	1	10	10
3	4	8	8

Trace

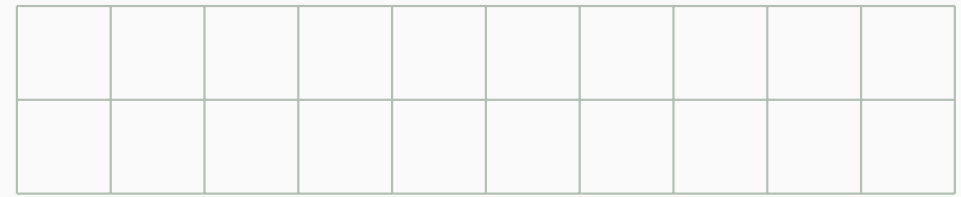
Tâches

#	est	lct	e
1	0	4	2
2	1	6	10
3	4	8	8

Trace

Tâches

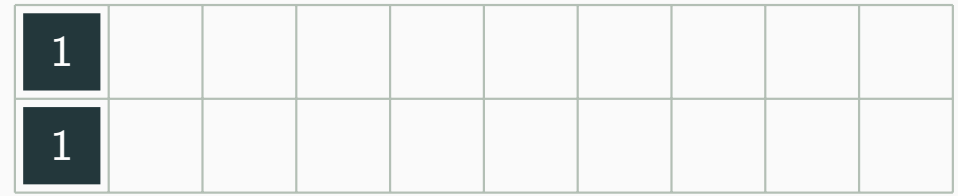
#	est	lct	e
1	0	4	2
2	1	6	10
3	4	8	8

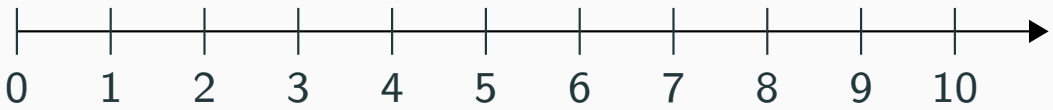


Trace

Tâches

#	est	lct	e
1	**0**	**4**	**2**
2	1	6	10
3	4	8	8

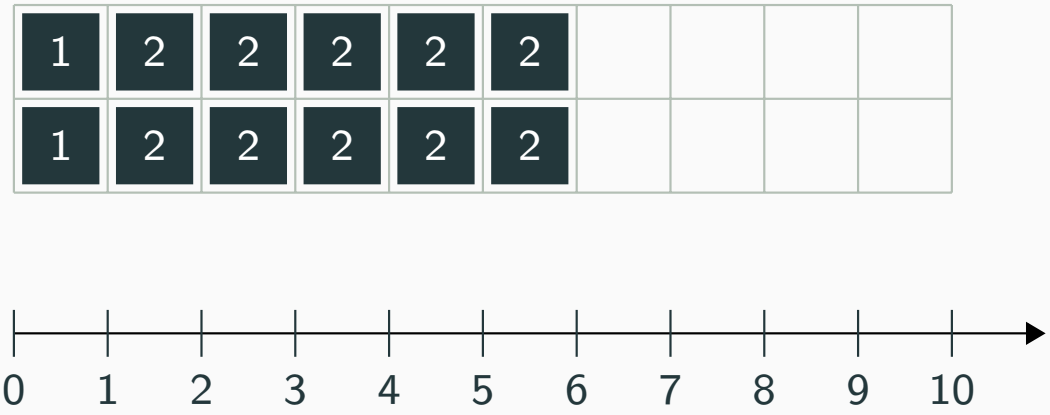




Trace

Tâches

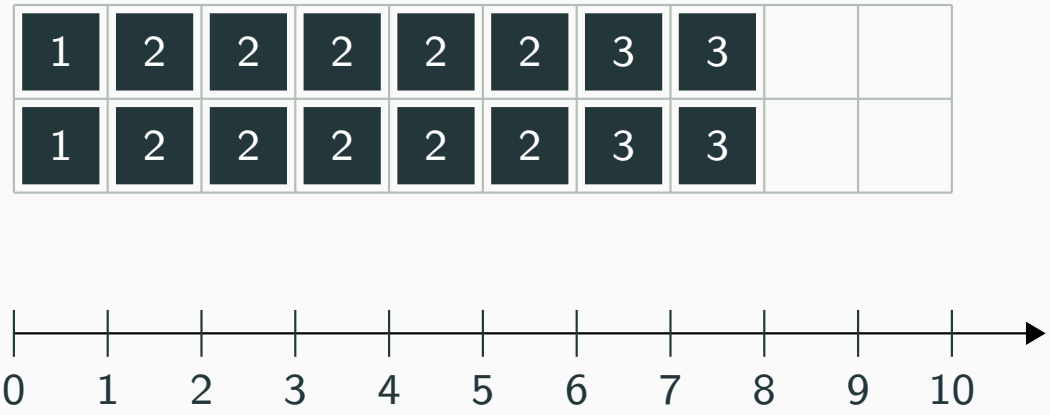
#	est	lct	e
1	0	4	2
2	**1**	**6**	**10**
3	4	8	8



Trace

Tâches

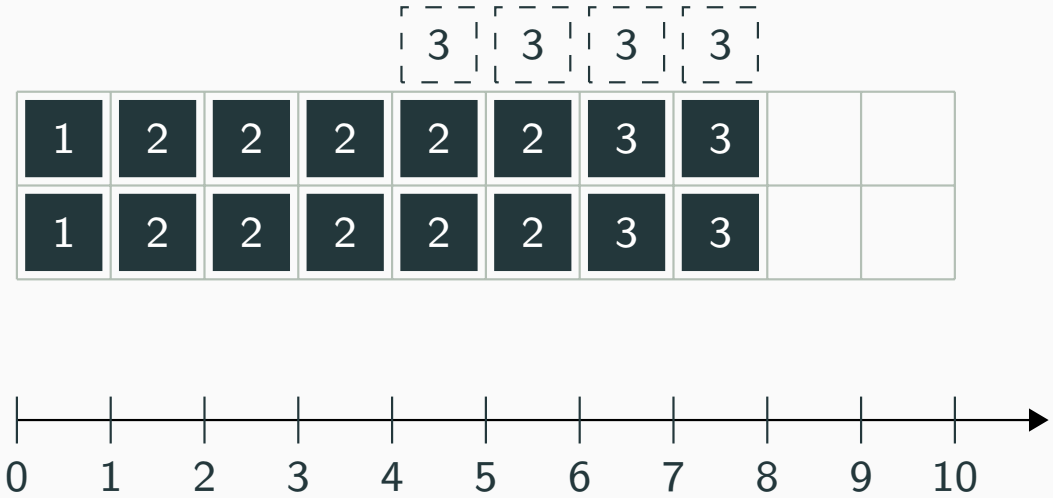
#	est	lct	e
1	0	4	2
2	1	6	10
3	**4**	**8**	**8**



Trace

Tâches

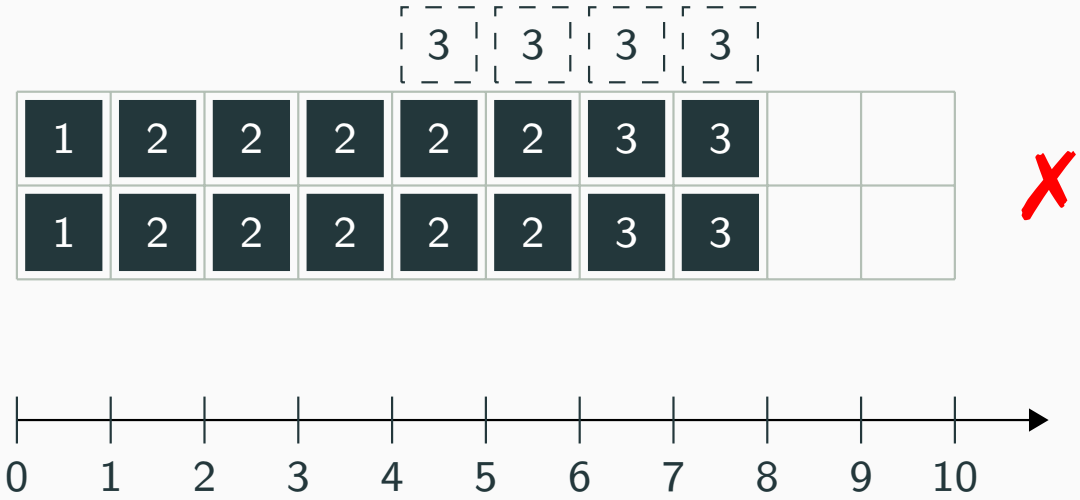
#	est	lct	e
1	0	4	2
2	1	6	10
3	**4**	**8**	**8**



Trace

Tâches

#	est	lct	e
1	0	4	2
2	1	6	10
3	**4**	**8**	**8**



Algorithme de filtrage - Complexité

Algorithme naïf

- Complexité dépend du nombre d'instants dans l'horizon

Algorithme de filtrage - Complexité

Algorithme naïf

- Complexité dépend du nombre d'instants dans l'horizon
- En pire cas, $\Theta(n^2 \cdot |T|)$

Algorithme de filtrage - Complexité

Algorithme naïf

- Complexité dépend du nombre d'instants dans l'horizon
- En pire cas, $\Theta(n^2 \cdot |T|)$

Algorithme intelligent

- Utilise l'overflow pour filtrer de plus qu'une unité
  - Détails dans l'article

Algorithme de filtrage - Complexité

Algorithme naïf

- Complexité dépend du nombre d'instants dans l'horizon
- En pire cas, $\Theta(n^2 \cdot |T|)$

Algorithme intelligent

- Utilise l'overflow pour filtrer de plus qu'une unité
  - Détails dans l'article
- Même pire cas, mais plus efficace en pratique

Expérimentations - Problème

Problème

Céduler des quarts de travail

Expérimentations - Problème

Problème

Céduler des quarts de travail

Employé

- Intervalle d'heures total à travailler
- Un quart maximum par jour
- 5 jours consécutifs maximum
- Entre 4 et 6 heures par quart

Expérimentations - Problème

Problème

Céduler des quarts de travail

Employé

- Intervalle d'heures total à travailler
- Un quart maximum par jour
- 5 jours consécutifs maximum
- Entre 4 et 6 heures par quart

Demande

Possiblement différente pour chaque tranche de 15 minutes

Expérimentations - Modèles

Concurrents

- Décomposition (en programmation par contraintes)
- Modèle MIP

Expérimentations - Modèles

Concurrents

- Décomposition (en programmation par contraintes)
- Modèle MIP

Objectif

- Avec un modèle « simple », trouver des solutions aux instances difficiles
- Être plus efficace que les autres méthodes exactes

Expérimentations - Modèles

Choix d'algorithme

- Vérification seulement
- Filtrage naïf
- Filtrage intelligent

Expérimentations - Modèles

Choix d'algorithme

- Vérification seulement
- Filtrage naïf
- Filtrage intelligent

Modélisation

- Une seule `MinCumulative` pour tout le problème
- Une `MinCumulative` par paire de jours
  - Un quart de travail qui commence au jour d doit terminer au plus au jour $d + 1$
  - Plus petit n pour chaque contrainte

Expérimentations

Méthodologie

- Modélisation en Minizinc
- Solveur de contraintes : Chuffed
- Solveur MIP : CPlex
- Limite de 20 minutes

Instances aléatoires

Description

- 400 instances générées pour comparer le comportement sur des instances faciles
- Paramètres générés aléatoirement avec un petit nombre de jours

Résultats - Instances aléatoires

Configuration	# Trouvée	# Meilleure
Décomposition	400	81
MIP	347	168
Vérification - Unique	398	94
Vérification - Multiple	400	236
Naïf - Unique	400	129
Naïf - Multiple	400	249
Intelligent - Unique	400	131
Intelligent - Multiple	400	**257**

- Trouvée : Instances pour lesquelles au moins une solution a été trouvée
- Meilleure : Instances pour lesquelles la meilleure solution a été trouvée

Véritables instances

Description

- 225 instances de Curtois *et al.*

Véritables instances

Description

- 225 instances de Curtois *et al.*
- Version simplifiée du banc d'essai original

Véritables instances

Description

- 225 instances de Curtois *et al.*
- Version simplifiée du banc d'essai original
- Très difficile pour une méthode exacte

Véritables instances

Description

- 225 instances de Curtois *et al.*
- Version simplifiée du banc d'essai original
- Très difficile pour une méthode exacte
- Même trouver une seule solution est difficile

Résultats - Véritables instances

Configuration	# Trouvée	# Meilleure
Décomposition	0	0
MIP	0	0
Vérification - Unique	**8**	**6**
Vérification - Multiple	0	0
Naïf - Unique	4	0
Naïf - Multiple	1	1
Intelligent - Unique	4	1
Intelligent - Multiple	1	0

- Trouvée : Instances pour lesquelles au moins une solution a été trouvée
- Meilleure : Instances pour lesquelles la meilleure solution a été trouvée

Conclusion

- Introduction de la contrainte `MinCumulative`
- Algorithme de vérification linéaire
- Algorithme de filtrage quadratique
- Efficace en pratique