Résolution efficace des problèmes d'ordonnancement sous contraintes complexes

Des résultats théoriques vers les applications pratiques

Christian Artigues

LAAS-CNRS

École Jeunes Chercheurs du GDR RO

26/11/2021

subsection in toc subsection in toc subsection in toc

subsection in toc subsection in toc subsection in toc

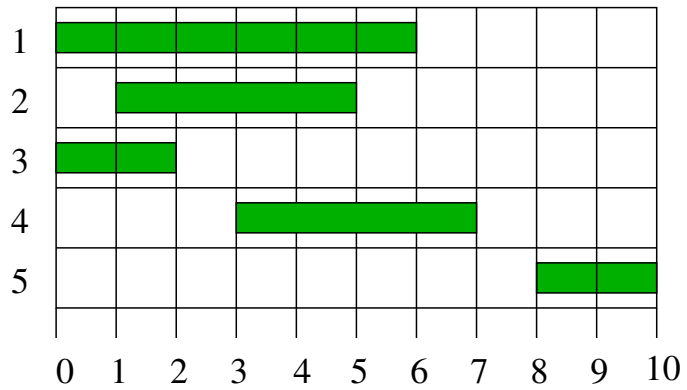
Le problème central de l'ordonnancement

Problème : Déterminer les dates de débuts d'un ensemble de n tâches $\mathcal{J}$ soumises à des contraintes de précédences Γ où $i \in \mathcal{J}$ est définie par

- Durée p_i
- Date de lancement r_i
- Deadline $\tilde{d}_i$
- Ensemble de successeurs Γ_i avec un délai $\delta_{i,j}$, $\forall j \in \Gamma_i$

Objectif : minimiser la durée du projet (plus grande date de fin des tâches)

i	p_i	r_i	$\tilde{d}_i$	Γ_i	$\delta_{i,j}, j \in \Gamma_i$
1	6	0	$+\infty$	2	1
2	4	0	$+\infty$	1	-3
3	2	0	3	$4, 5$	$2, 2$
4	4	0	$+\infty$	3	-4
5	2	8	$+\infty$	4	-5

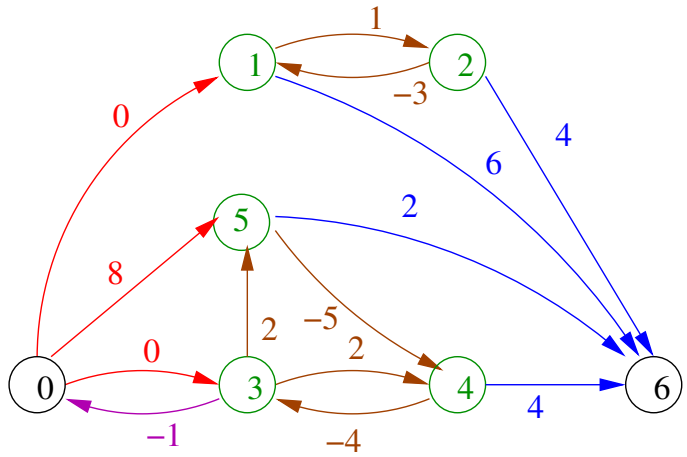


Le graphe conjonctif ou potentiels-tâches 1/2

On construit le graphe $G = (V, A, l)$ comme suit :

- Ensemble des sommets : $V = \mathcal{J} \cup \{0, n+1\}$
- Ensemble des arcs :
  $A = \{l(0,i), (i,0), (i,n+1) | \forall i \in \mathcal{J}\} \cup \{(i,j) | \forall i \in \mathcal{J}, \forall j \in \Gamma_i\}$
- Valuations des arcs : $\forall i \in \mathcal{J}$: $l_{0,i} = r_i$, $l_{i,0} = p_i - \tilde{d}_i$, $l_{i,n+1} = p_i$, et $l_{i,j} = \delta_{i,j}$, $\forall j \in \Gamma_i$

i	p_i	r_i	$\tilde{d}_i$	$\Gamma(i)$	$l_{ij}, j \in \sigma(i)$
1	6	6	$+\infty$	2	1
2	4	0	$+\infty$	1	-3
3	2	0	3	4, 5	2, 2
4	4	0	$+\infty$	3	-4
5	2	8	$+\infty$	4	-5



Le graphe conjonctif ou potentiels-tâches 2/2

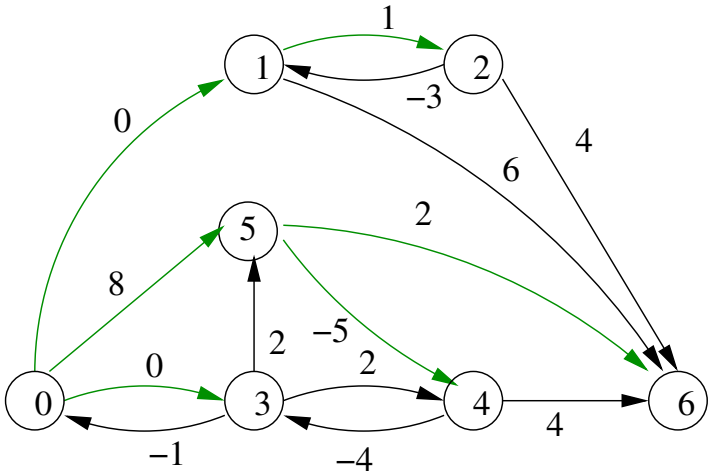
Propriétés remarquables

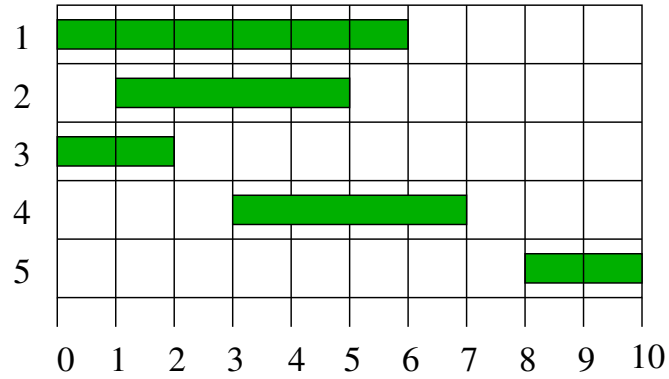
Soit $\mathcal{S}$ l'ensemble des ordonnancements réalisables

- Le problème est irréalisable $\mathcal{S} = \emptyset$ si et seulement s'il existe un circuit de longueur positive dans G
- La date de début au plus tôt $ES_i = \min_{S \in \mathcal{S}} S_i$ est le plus long chemin de 0 à i
- La date de début au plus tard $LS_i = \max_{S \in \mathcal{S} | S_{n+1} = ES_{n+1}} S_i$ est ES_{n+1} moins la longueur du plus long chemin de i à $n+1$

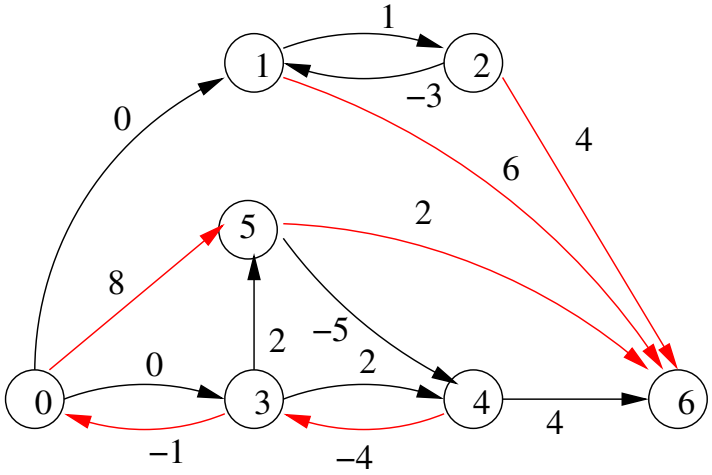
Ordonnancements au plus tôt/tard, chemin critique

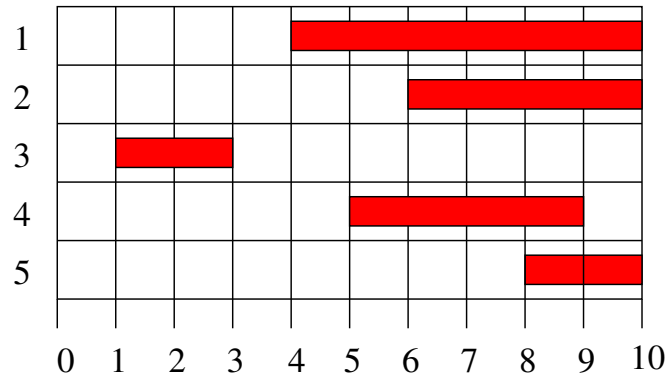
Ordonnancement au plus tôt et plus longs chemins correspondants





Ordonnancement au plus tard et plus longs chemins correspondants





Résolution

Formulation primale

$\min\ S_{n+1}$ s.c.
$S \in \mathcal{S} =$
$\{S_j - S_i \geq l_{i,j}, (i,j) \in E\}$

Problème de *potentiels* sur les sommets dans un graphe potentiel-tâches.

Formulation duale

$\max \sum_{(i,j)\in E} l_{i,j}\phi_{i,j}$ s.c.
$\Phi \in \mathcal{D} =$

$$\left\{ \begin{array}{l} \sum\limits_{(i,j)\in E} \phi_{i,j} = \sum\limits_{(i,j)\in E} \phi_{j,i}, i \in \{1,...,n\} \\ \sum\limits_{(i,n+1)\in E} \phi_{i,n+1} = 1 \\ \phi_{i,j} \geq 0, (i,j) \in E \cup (n+1,0) \end{array} \right\}$$

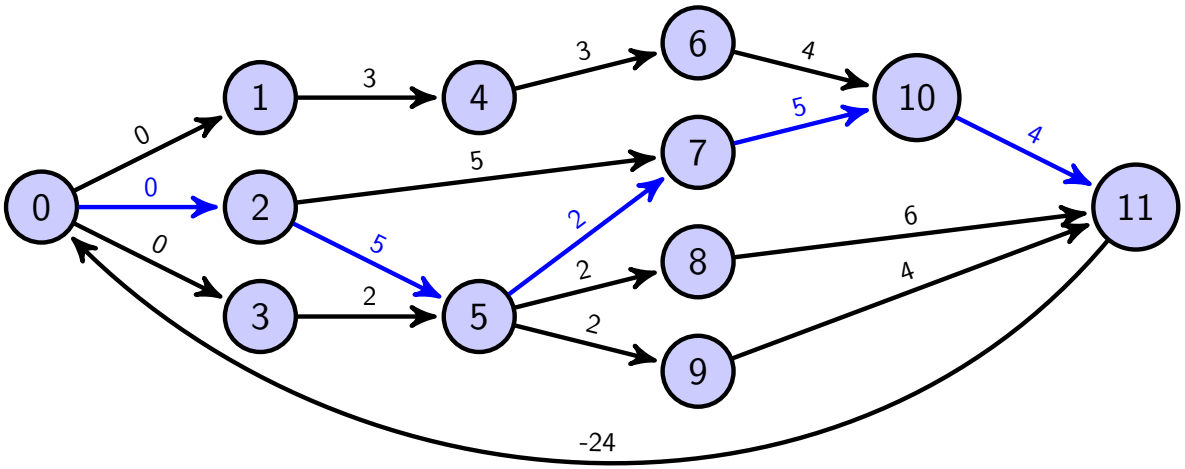
Problème de *plus long chemin* entre 0 et $n+1$.

- Formulation récurrente $S_0^0 = 0$, $S_j^0 = -\infty$, $j \in V \setminus \{0\}$ et $S_j^k = \max(S_j^{k-1}, \max_{i\in V|(i,j)\in A} S_i^{k-1} + l_{i,j})$, $j \in V$, $k \geq 1$

  Se résout par programmation linéaire ou bien par l'algorithme de Bellman-Ford en $O(|V||A|)$.

Cas particulier, le problème d'ordonnancement de projet

Lorsque on a $\delta_{i,j} = p_i$, $\forall (i,j) \in \Gamma$ et pas de deadlines, on parle d'ordonnancement de projet standard (CPM, PERT, MPM).
Dans ce cas, il ne peut y avoir de cycles dans le graphe et un tri topologique des sommets permet de faire la récursion en $O(|A|)$.

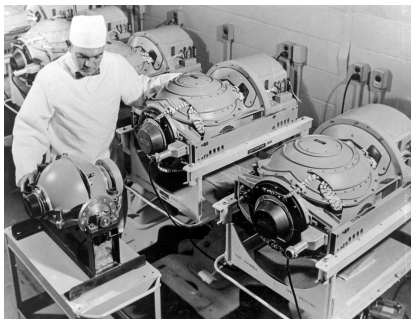


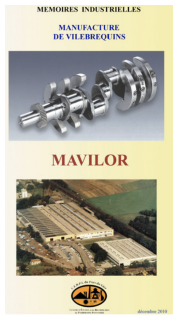
$$T = 24$$

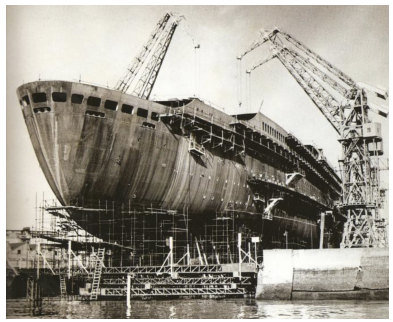
i	p_i	$[ES, LS]$
1	3	$[0, 10]$
2	5	$[0, 8]$
3	1	$[0, 12]$
4	3	$[3, 13]$
5	2	$[5, 13]$
6	4	$[6, 16]$
7	5	$[7, 15]$
8	6	$[7, 18]$
9	4	$[7, 20]$
10	4	$[12, 20]$
11	0	$[16, 24]$

Un peu d'histoire et d'applications prestigieuses

- Méthode CPM (Critical Path Method). Inventée par James E. Kelley (Remington Rand) et Morgan R. Walker entre 1956 et 1959 (Dupond). Utilisée en 1966 pour la construction des tours jumelles du World Trade Center.
- Méthode PERT (Program Evaluation & Review Technique) Inventée par Booz Allen Hamilton et la U.S. Navy pour le développement des missiles Polaris en 1958
- Méthode MPM (Méthodes des Potentiels Métra). Inventée par Bernard Roy en 1958 pour la société Sema et la planification de la production de villebrequins de l'Usine MAvilor puis de l'équipement du paquebot France.







Extensions, pour aller plus loin

- Incertitudes

  Ernst Roos, Dick den Hertog (2021) A distributionally robust analysis of the program evaluation and review technique, European Journal of Operational Research, Volume 291, Issue 3, Pages 918-928

- Controlabilité

  Gao, M., Popowski, L., Boerkoel, J. (2020). Dynamic Control of Probabilistic Simple Temporal Networks. Proceedings of the AAAI Conference on Artificial Intelligence, 34(06), 9851-9858.

- Autres fonctions de coûts

  Chrétienne, P., Sourd, F. (2003). PERT scheduling with convex cost functions. Theoretical Computer Science, 292(1), 145-164.

- Activity Crashing
- Gestion des ressources

  à suivre....

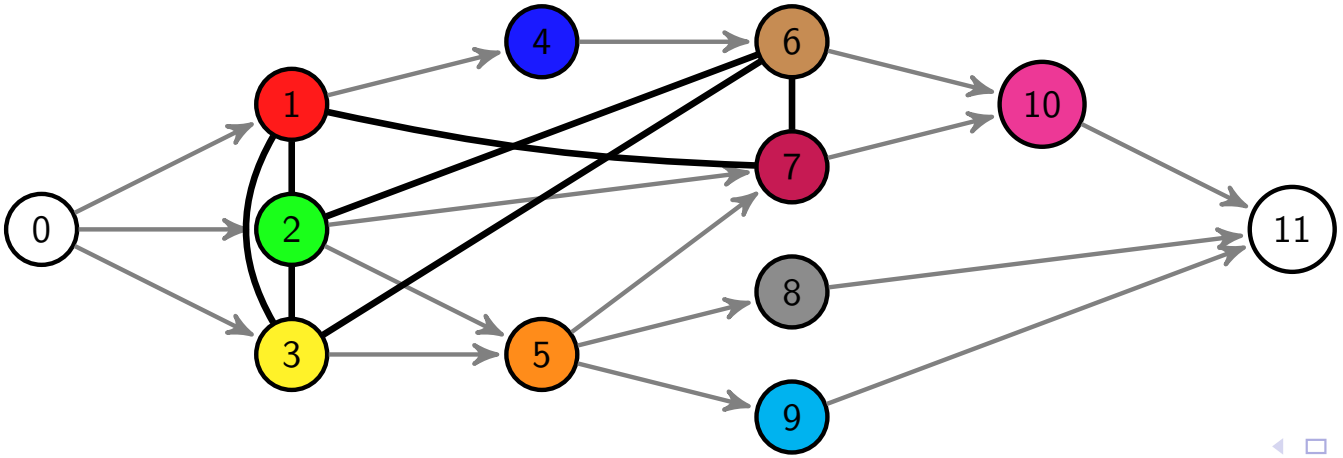
subsection in toc subsection in toc subsection in toc

Les problèmes d'ordonnancement à ressources non partageables : le graphe disjonctif

- La contrainte disjonctive exprime le non parallélisme entre deux tâches

$$S_j \geq S_i + p_i \vee S_i \geq S_j + p_j$$

- En général les tâches utilise une ressource non partageable
- D ensemble des paires de tâches en disjonction.
- Extension du graphe conjonctif : ajout d'une arête $\{i, j\}$ pour chaque paire de disjonction $\{i, j\} \in D$.



$$D = \{\{1,2\},\{1,3\},\{2,3\}$$
$$\{2,6\},\{3,6\},\{1,7\},\{6,7\}\}$$

Le graphe disjonctif : propriété fondamentale

Définition

Un ordonnancement S est semi-actif si pour tout $\epsilon > 0$ et toute tâche $i \in \mathcal{J}$, l'ordonnancement S' tel que $S'_i = S_i - \epsilon$ et $S'_j = S_i$, $\forall j \in \mathcal{J} \setminus \{i\}$ est irréalisable.

Propriété

L'ensemble des ordonnancement semi-actif est dominant pour toute fonction objectif régulière (c'est à dire non décroissante en fonction des dates de fin des tâches (e.g $\max_{j \in \mathcal{J}} S_j + p_j$, $\sum_{j \in \mathcal{J}} S_j + p_j$, ...)

Théorème

Il existe une bijection entre l'ensemble des ordonnancements semi-actifs et l'ensemble des orientations acycliques du graphe disjonctif

Formulation du problème

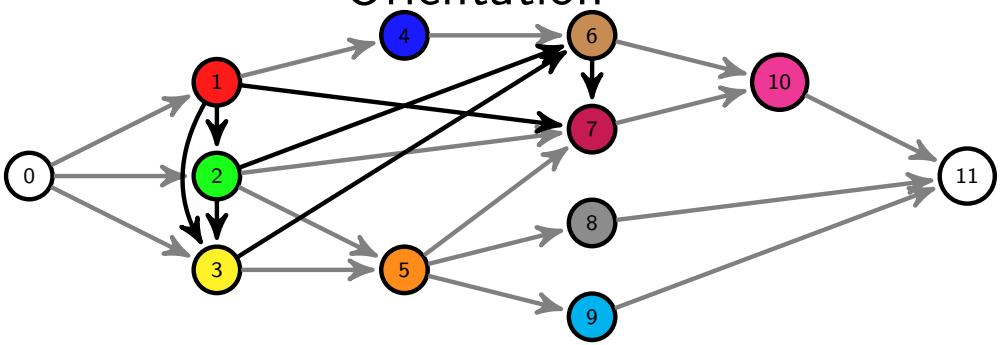
Trouver une orientation acyclique Π du graphe disjonctif qui minimise la longueur du plus long chemin entre 0 et $n+1$ (NP-difficile au sens fort). Équivaut à résoudre ce PLNE à temps continu :

$$
\begin{aligned}
\min\ & S_{n+1} & \\
\text{s.c. } & S_j \geq S_i + l_{i,j} & \forall (i,j) \in E \\
& S_j \geq S_i + p_i + (1 - M_{ij}) x_{ij} & \forall \{i,j\} \in D \\
& S_i \geq S_j + p_j + (1 - M_{ji}) x_{ji} & \forall \{i,j\} \in D \\
& x_{ij} + x_{ji} = 1 & \forall \{i,j\} \in D \\
& S_i \geq 0 & \forall i \in V \\
& x_{i,j} \in \{0,1\} & \forall \{i,j\} \in D
\end{aligned}
$$

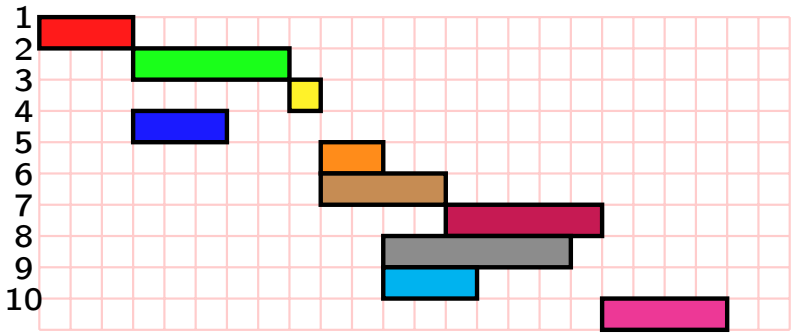
avec $M_{ij} - p_i$ une borne supérieure valide de $S_i - S_j$

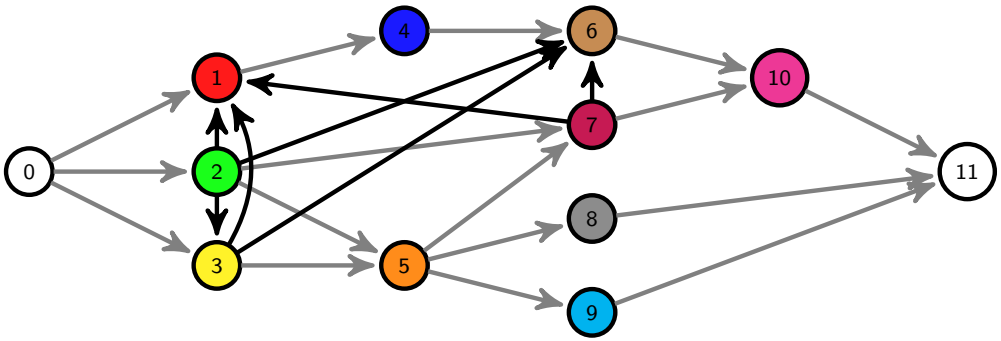
Exemples d'orientations réalisables et irréalisables

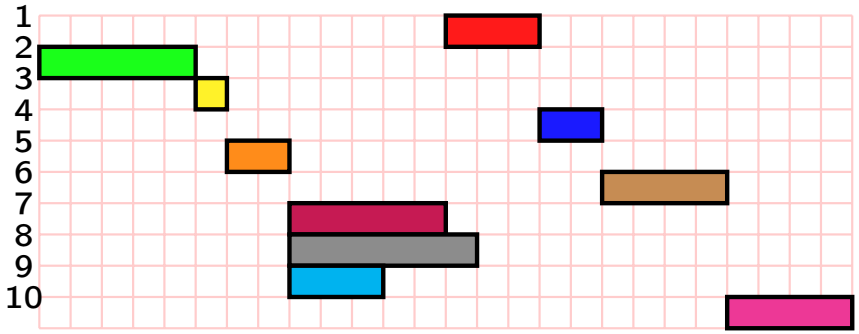
Orientation

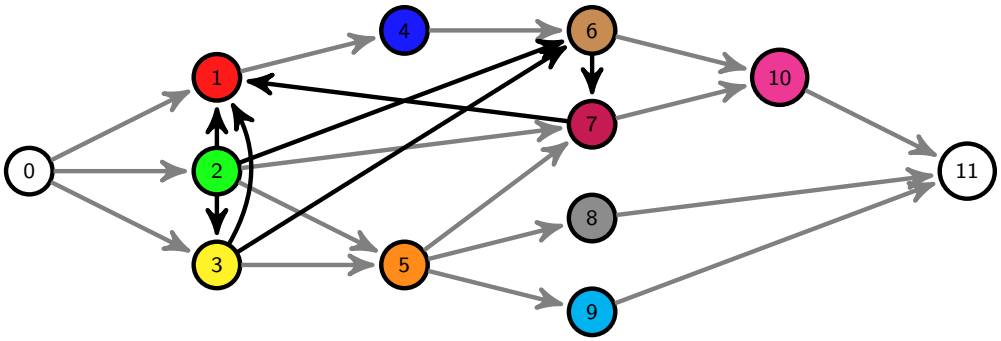


Ordonnancement au plus tôt







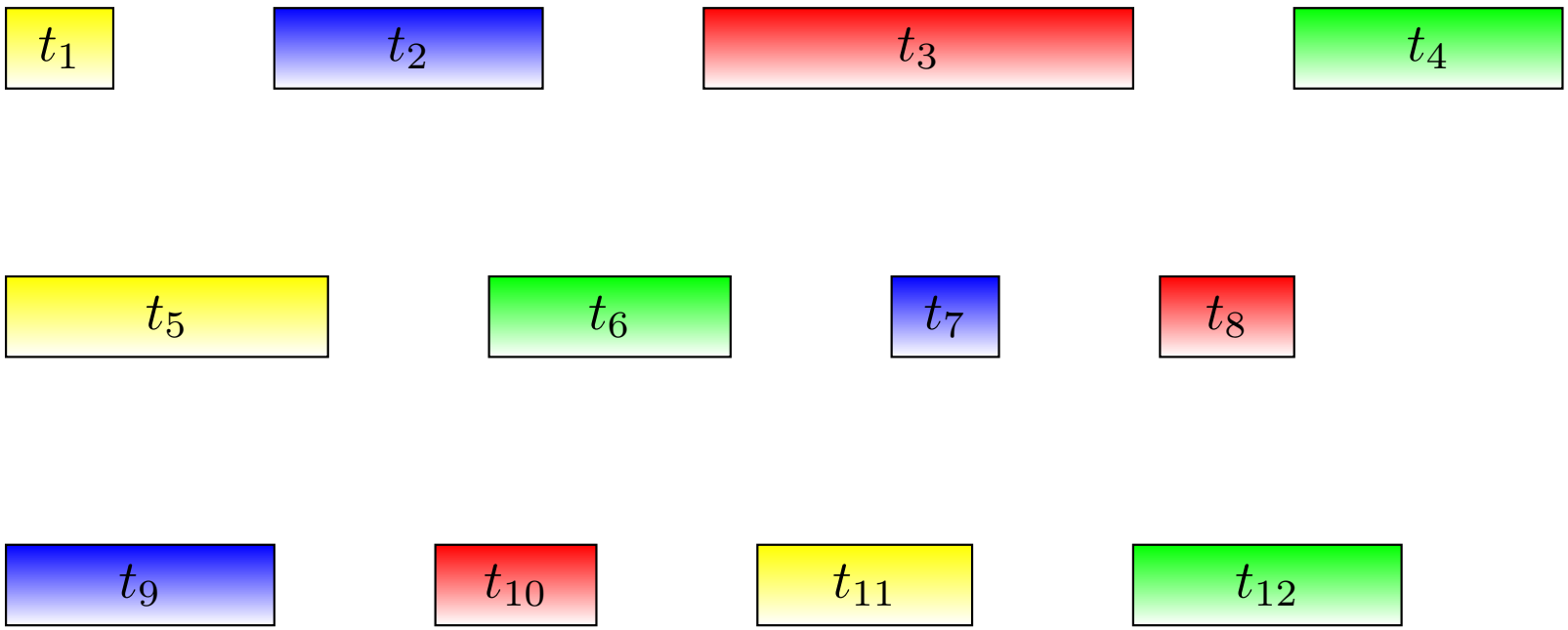


IRRÉALISABLE

subsection in toc subsection in toc subsection in toc

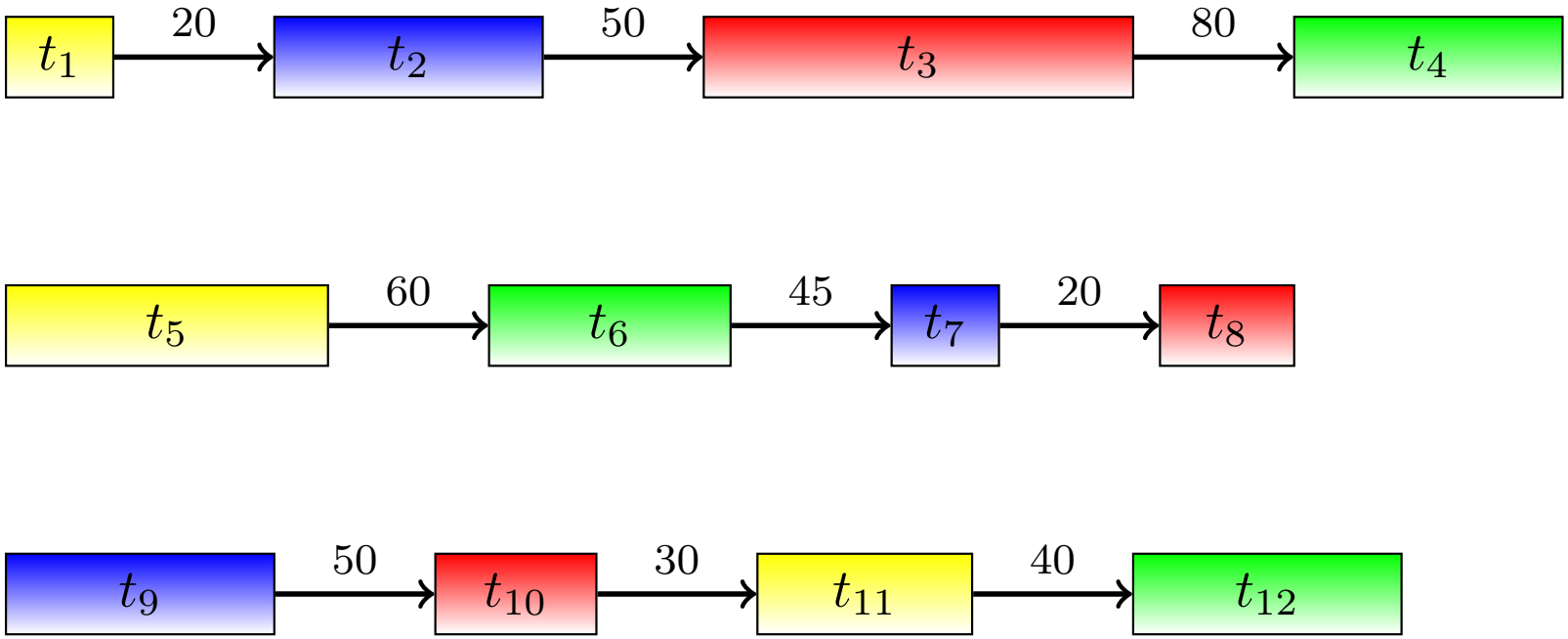
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



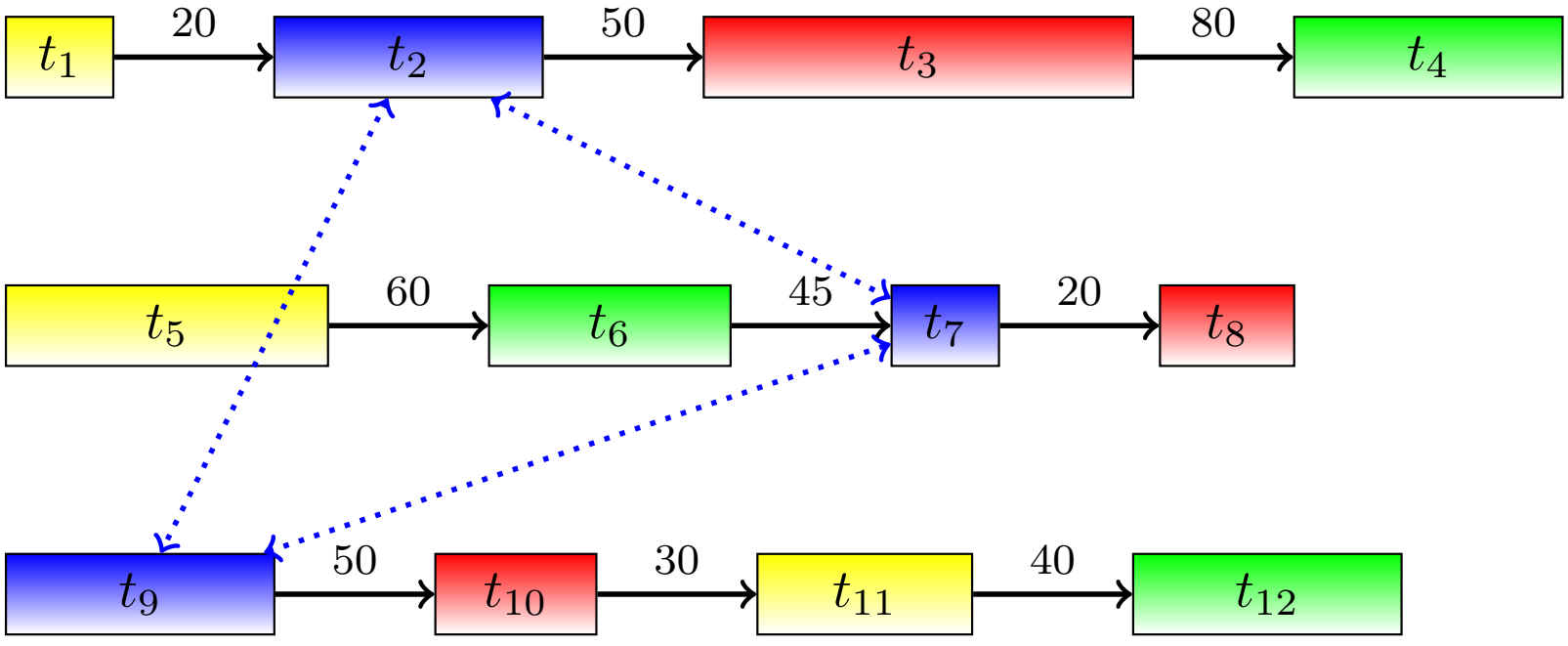
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



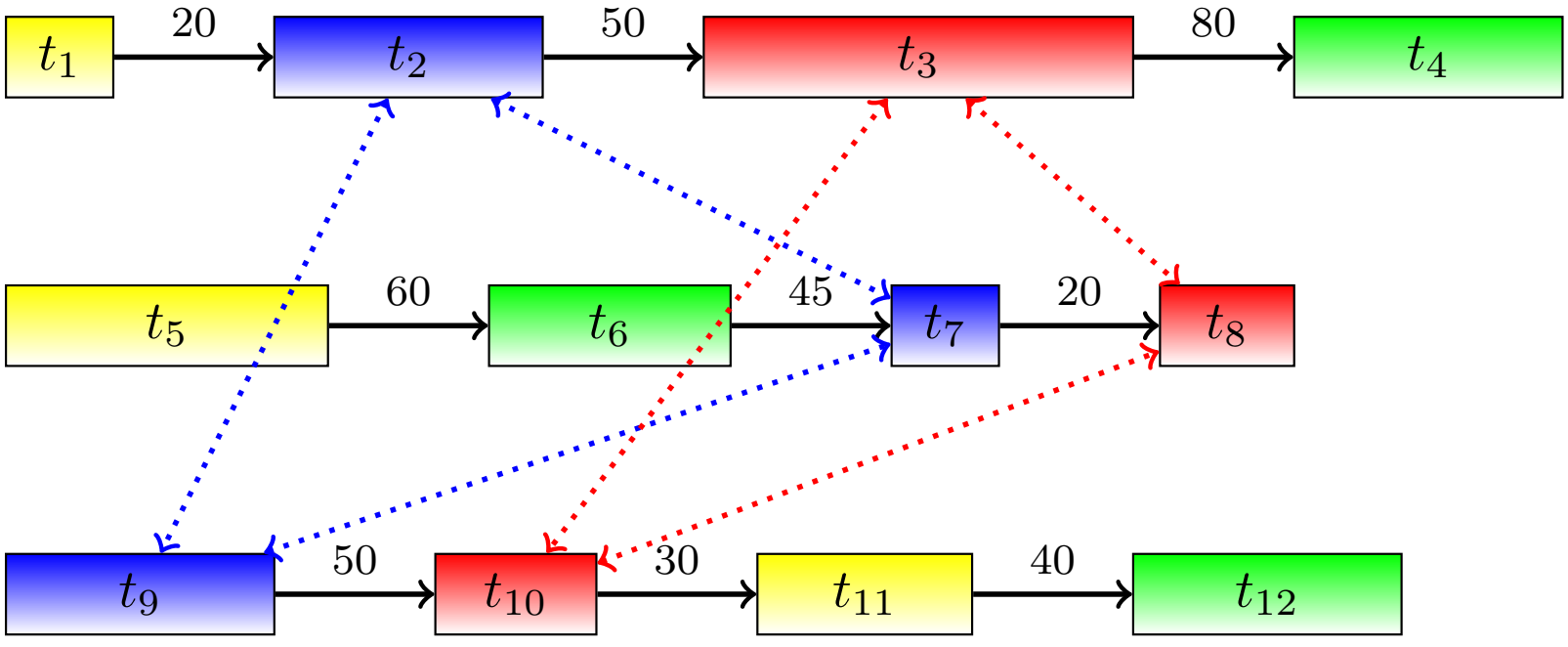
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



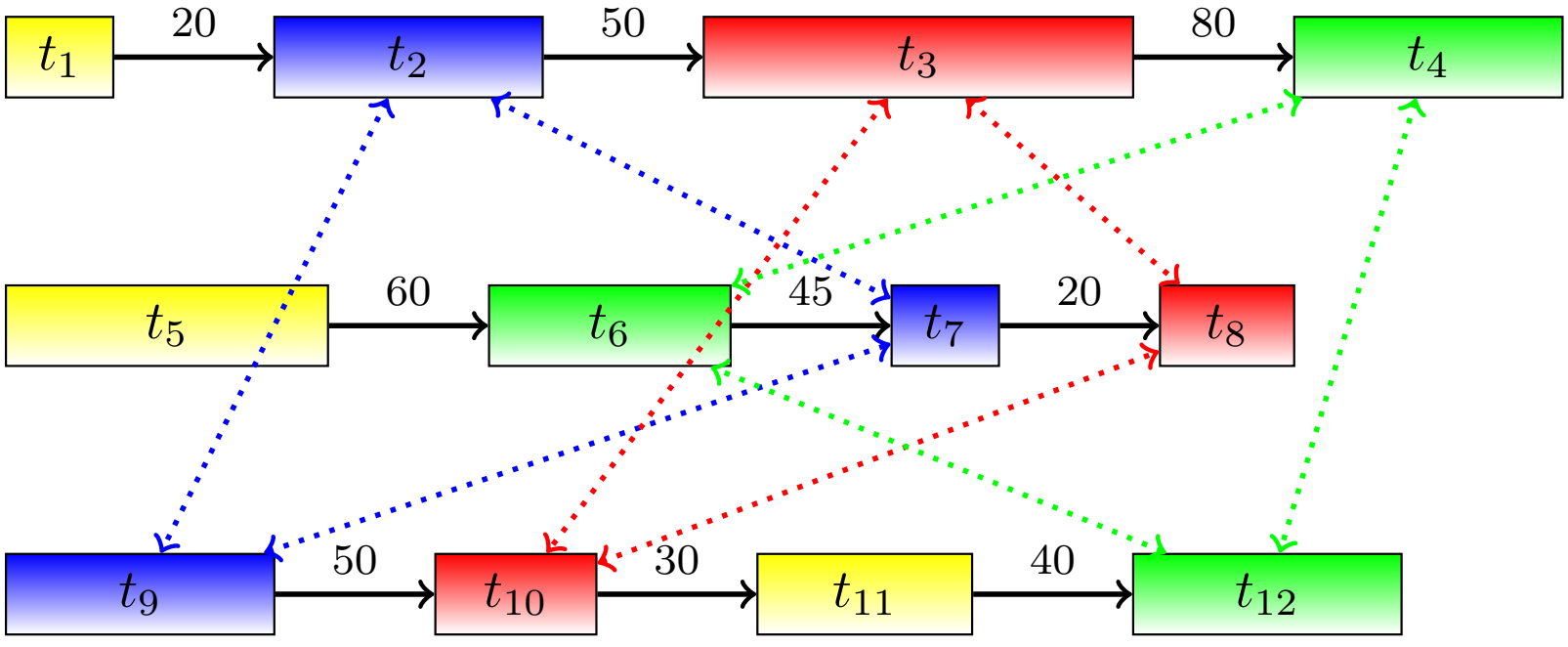
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



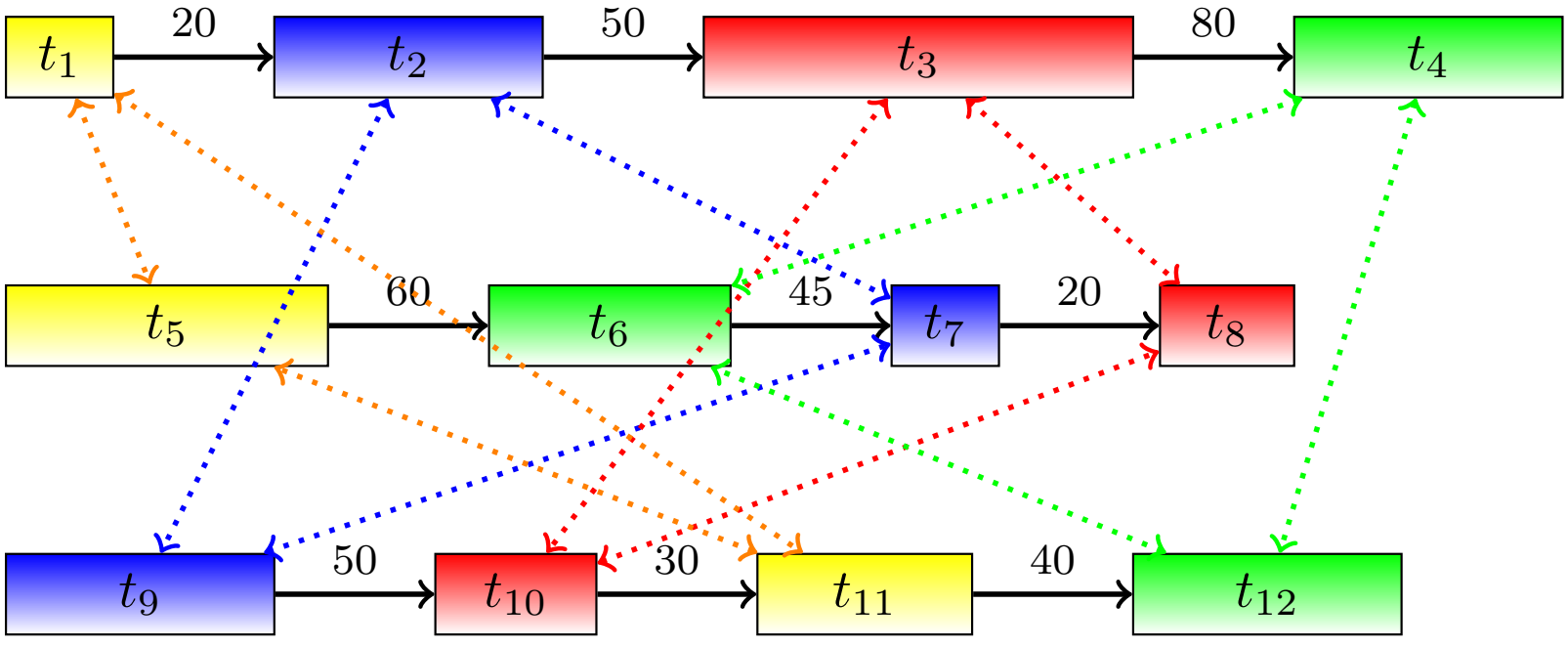
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



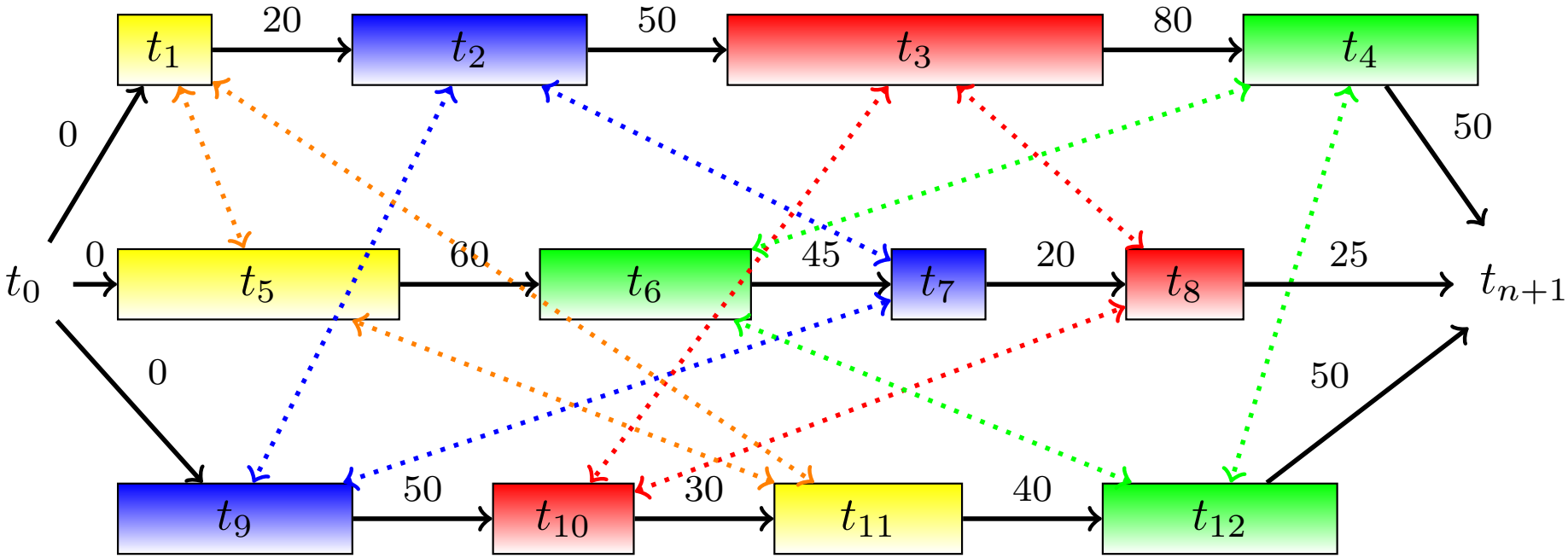
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



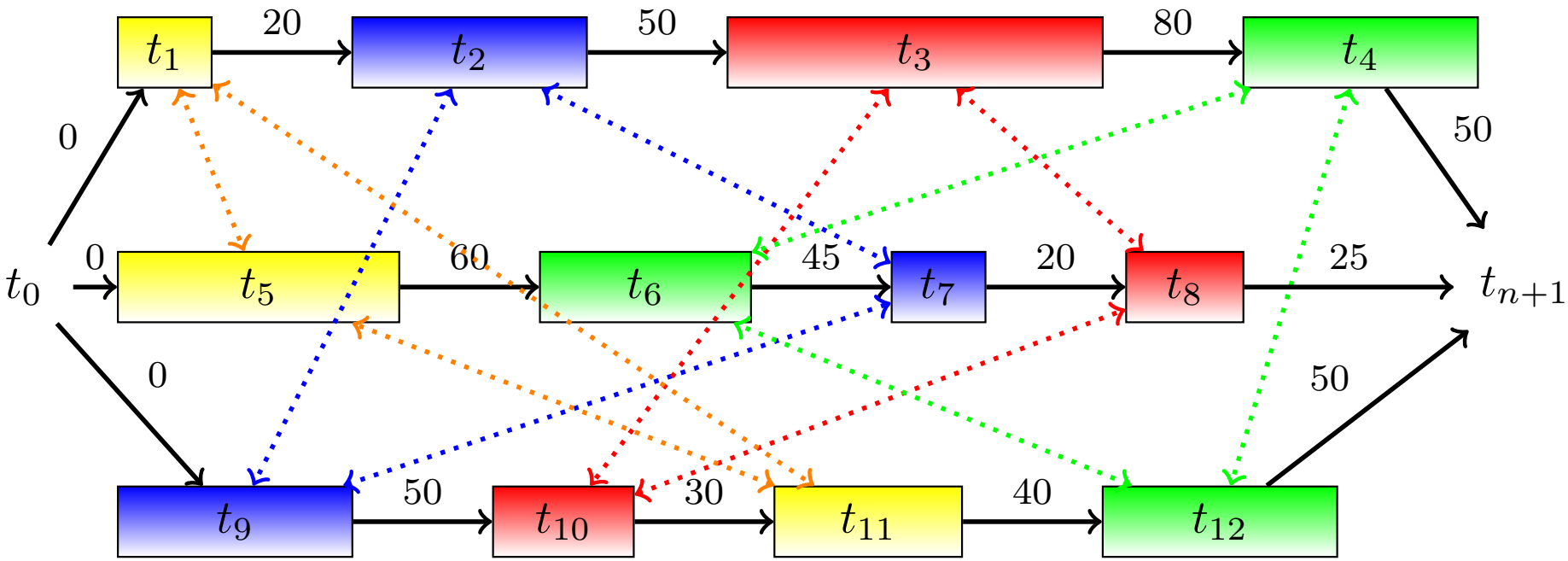
Exemple classique : le problème du Job-shop

N travaux décomposés en tâches, m machines $(N = 3, m = 4)$



Exemple classique : le problème du Job-shop

n travaux décomposés en tâches, m machines $(n = 3, m = 4)$



$\mathcal{J}_k$: ensemble des tâches exécutées sur machine k : clique maximale du graphe des disjonctions.

Borne des cliques de disjonctions

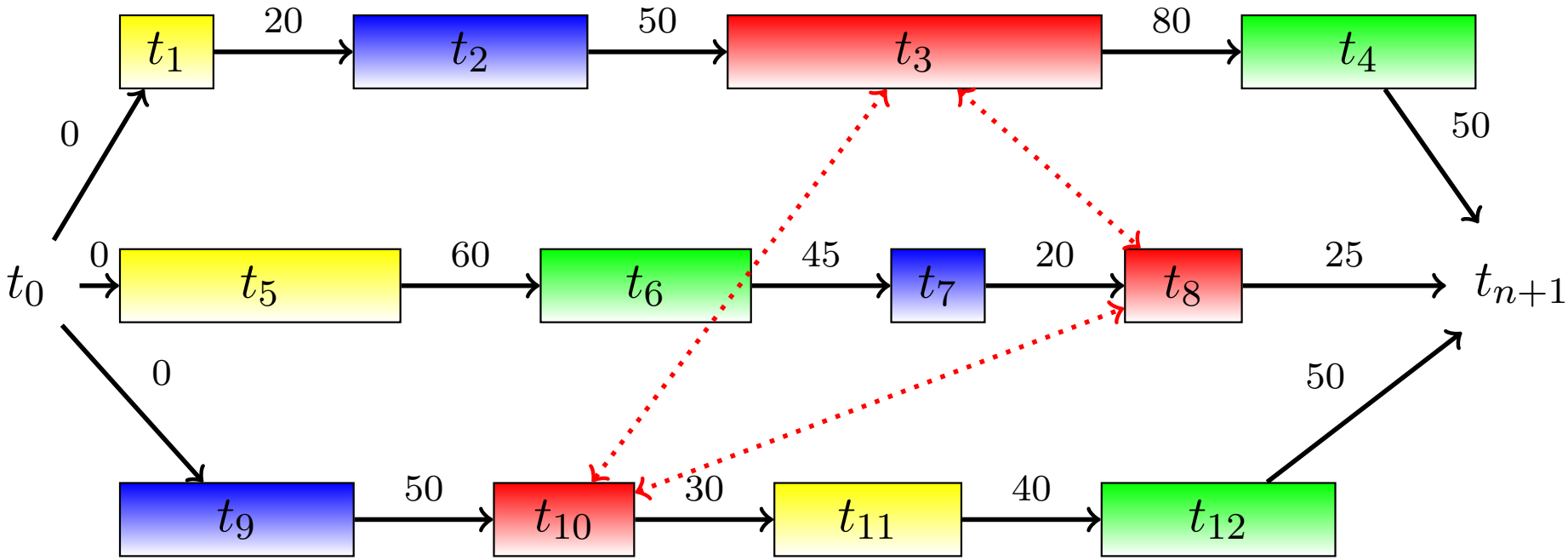
Soit $l^*_{i,j}$ la longueur du plus long chemin entre $i \in V$ et $j \in V$ dans le graphe disjonctif partiellement orienté.
Pour une tâche donnée

- $r_i = l^*_{0,i}$ est la durée minimale entre le début de l'ordonnancement et le début de i
- $q_i = l^*_{i,n+1} - p_i$ est la durée minimale entre la fin de i et la fin de l'ordonnancement

C clique du graphe des disjonctions (sous-ensemble de tâches, non nécessairement maximal, exécutées sur la même machine)
Une borne inférieure associée à toute clique C est donc

$$h(C) = \min_{i \in C} r_i + \sum_{i \in C} p_i + \min_{i \in C} q_i$$

Borne des cliques de disjonctions



exemples :

$h(\{t_3, t_8, t_{10}\}) = \min(20 + 50, 60 + 45 + 20, 50\} + 80 + 25 + 30 + \min\{50, 0, 40 + 50\} = 50 + 135 + 0 = 185$

$h(\{t_3, t_{10}\}) = \min(20 + 50, 50\} + 80 + 30 + \min\{50, 40 + 50\} = 50 + 110 + 50 = 210$

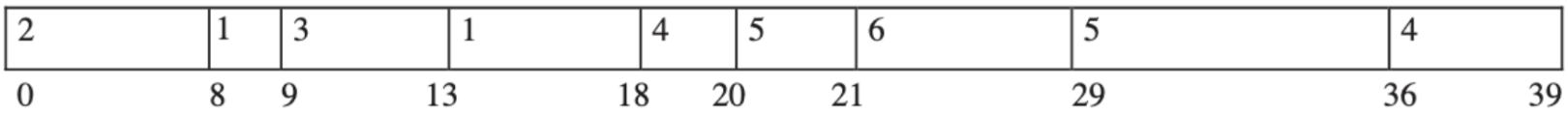
Bornes des cliques et problème à une machine préemptif

Etant donné une clique maximale $\mathcal{J}_k$, comment calculer la meilleure borne $h^*(\mathcal{J}_k) = \max_{C \subseteq \mathcal{J}_k} h(\mathcal{J}_k)$?

Carlier [1982] prouve que la résolution du problème à une machine préemptif $1|r_i, q_i, pmtn|C_{\max}$ donne $h^*(\mathcal{J}_k)$

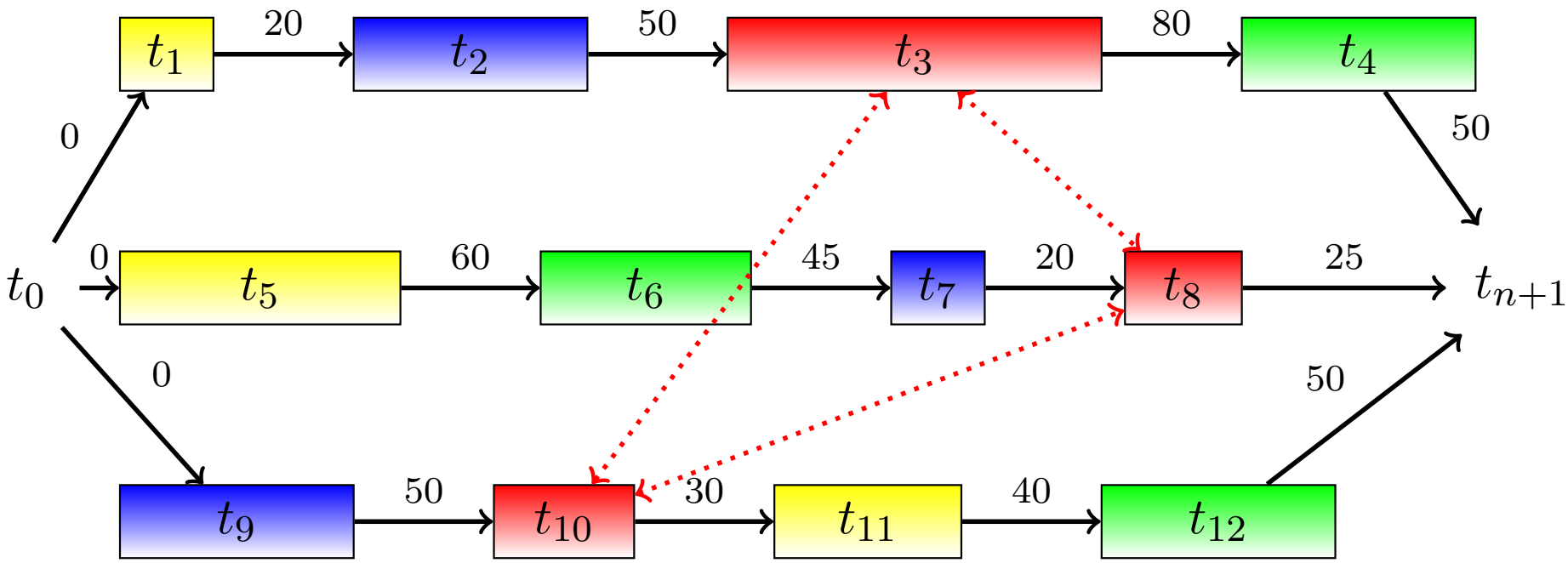
L'algorithme de Jackson Préemptif (JPS) résout le problème pour chaque machine en $O(nlogn)$

	1	2	3	4	5	6
r_i	4	0	9	15	20	21
p_i	6	8	4	5	8	8
q_i	20	25	30	9	14	16



subsection in toc subsection in toc subsection in toc

Le problème à une machine non préemptif



Une borne plus forte peut être obtenue en résolvant le problème non préemptif $1|r_i, q_i|C_{\max}$ (NP-difficile)

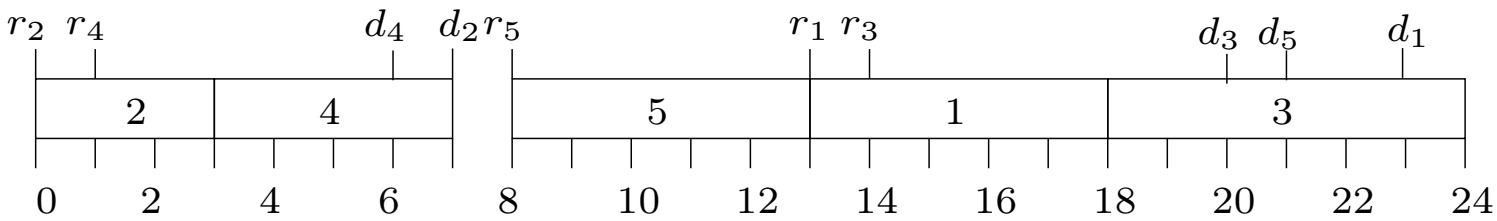
Remarque : équivalent à $1|r_i|L_{\max}$ en posant $d_i = T - q_i$ avec T constante arbitraire (exemple $T = \max_{i \in \mathcal{J}_{\|}} q_i$)

Une heuristique pour le $1|r_i|L_{\max}$

Une borne supérieure du problème peut être obtenu par l'algorithme de Jackson Non Préemptif (JNPS)

1. $t \leftarrow \min_{i \in T} r_i$
2. Sélectionner la tâche i telle que $r_i \leq t$ de plus petite date de livraison et l'ordonnancer à t.
3. décaler t à la première date où une tâche non-ordonnancée est disponible et aller à l'étape 2.

i	p_i	d_i
1	5	23
2	3	7
3	6	20
4	4	6
5	5	21



solution par l'algorithme de Jackson non préemptif $L_{\max} = 4$

Rq: La borne inférieure de cliques devient $h(C) = \min_{i \in C} r_i + \sum_{i \in C} p_i - \max_{i \in C} d_i$

Un résultat d'approximation utile en pratique !

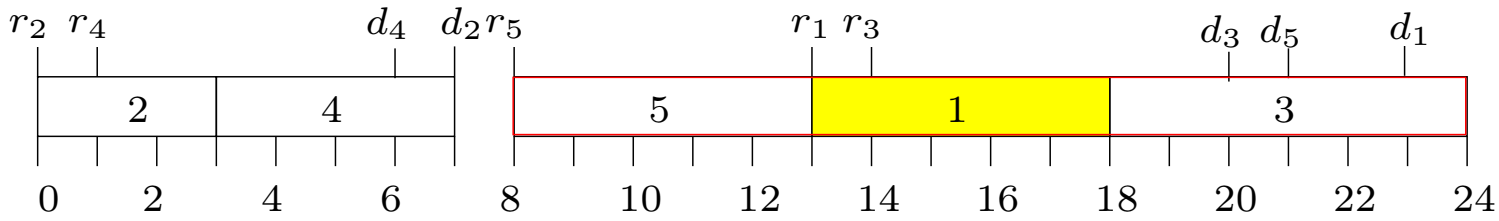
Soit $i_1, \ldots, i_p$ le "chemin critique" tel que $UB = r_1 + \sum_{k=1,\ldots,p} p_{i_k} - d_{i_p}$.

Si pour tout $k = 1, \ldots, p-1$, $d_{i_k} \leq d_{i_p}$, la solution est optimale.

Sinon, soit c le plus grand indice tel que $d_{i_c} > d_{i_p}$. Soit de plus $J = \{i_c + 1, \ldots, i_p\}$

Théorème [Carlier 1982]

Dans toute solution optimale i_c est exécuté, soit avant, soit après toutes les tâches de J. L'écart à l'optimum de la solution JNPS est d'au plus p_{i_c}.



Bloc critique $\{5, 1, 3\}$ tâche critique 1 $J = \{3\}$

Un résultat d'approximation utile en pratique !

Preuve :

$i_1, \ldots, i_p$ le "chemin critique"

c le plus grand indice tel que $d_{i_c} > d_{i_p}$

$J = \{i_c + 1, \ldots, i_p\}$

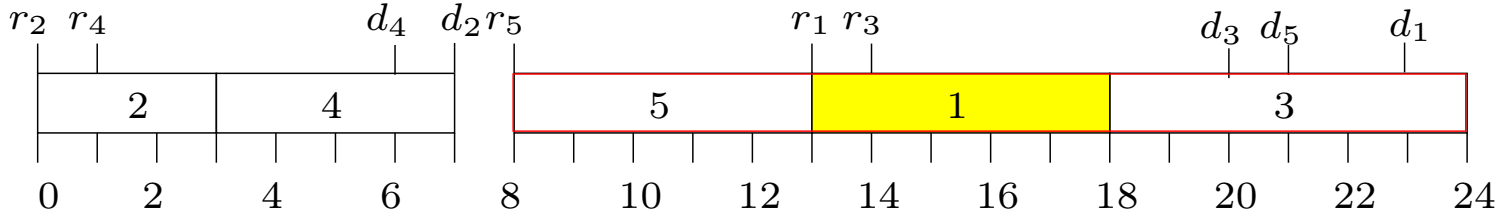
$S_{i_c} = r_{i_1} + p_{i_1} + ... + p_{i_{c-1}} < \min_{j \in J} r_j$ (sinon une tâche de J aurait été ordonnancée à la place de i_c)

or $d_{i_p} = \max_{j \in J} d_j$

D'où $r_{i_1} + p_{i_1} + ...p_{i_p} - d_{i_p} - p_{i_c} < \min_{j \in J} r_j + p_{i_{c+1}} + ... + p_{i_p} - \max_{j \in J} d_j$

ou encore $UB = r_{i_1} + p_{i_1} + ...p_{i_p} - d_{i_p} < \min_{j \in J} r_j + p_{i_{c+1}} + ... + p_{i_p} + p_{i_c} - \max_{j \in J} d_j$

i_c est non insérable dans J et l'écart à l'optimum de UB est inférieur à p_{i_c}



Bloc critique $\{5, 1, 3\}$ tâche critique 1 $J = \{3\}$

Résolution exacte du problème à une machine non préemptif

Algorithme de Carlier, 1982

1. $BS \leftarrow +\infty$
2. $Q \leftarrow$Empiler(r, p, d)
3. Tant que $Q \neq \emptyset$ Faire
4. $\quad (r, p, d) =$Dépiler(Q)
5. $\quad$Calculer i_c et J par JNP et Mettre à jour BS si nécessaire.
6. $\quad$Si la condition d'optimalité est vérifiée retourner JNP.
7. $\quad$Calculer BI par l'algorithme de Jackson préemptif.
8. $\quad$Si $BI \leq BS$ alors
9. $\quad\quad r' \leftarrow r$; $r'_{i_c} \leftarrow \max(r_{i_c}, \min_{i \in J} r_i + \sum_{i \in J} p_i)$
10. $\quad\quad d' \leftarrow d$; $d'_{i_c} \leftarrow \min(d_{i_c}, \max_{i \in J} d_i - \sum_{j \in J} p_i)$.
11. $\quad\quad Q \leftarrow$Empiler(r, p, d')
12. $\quad\quad Q \leftarrow$Empiler(r', p, d)

Agorithme de Carlier (1982) : résultats et conséquences

1. L'algorithme est extrêmement efficace. Résolution quelques secondes d'instances avec des milliers de tâches.
2. Le graphe disjonctif offre un outil de modélisation et un support algorithmique puissant.

n	T_{Car}		
	min	mean	max
100	0	0.02	1
400	0	0.3	2
700	0	1.4	6
1000	0	1.22	27
1300	0	2.46	20
1600	0	5.48	29
1900	0	3.42	40
2200	0	1.06	20
2500	0	1.9	31
2800	0	0.28	1
3100	0	0.4	1

Table : [Briand *et al* 2010]

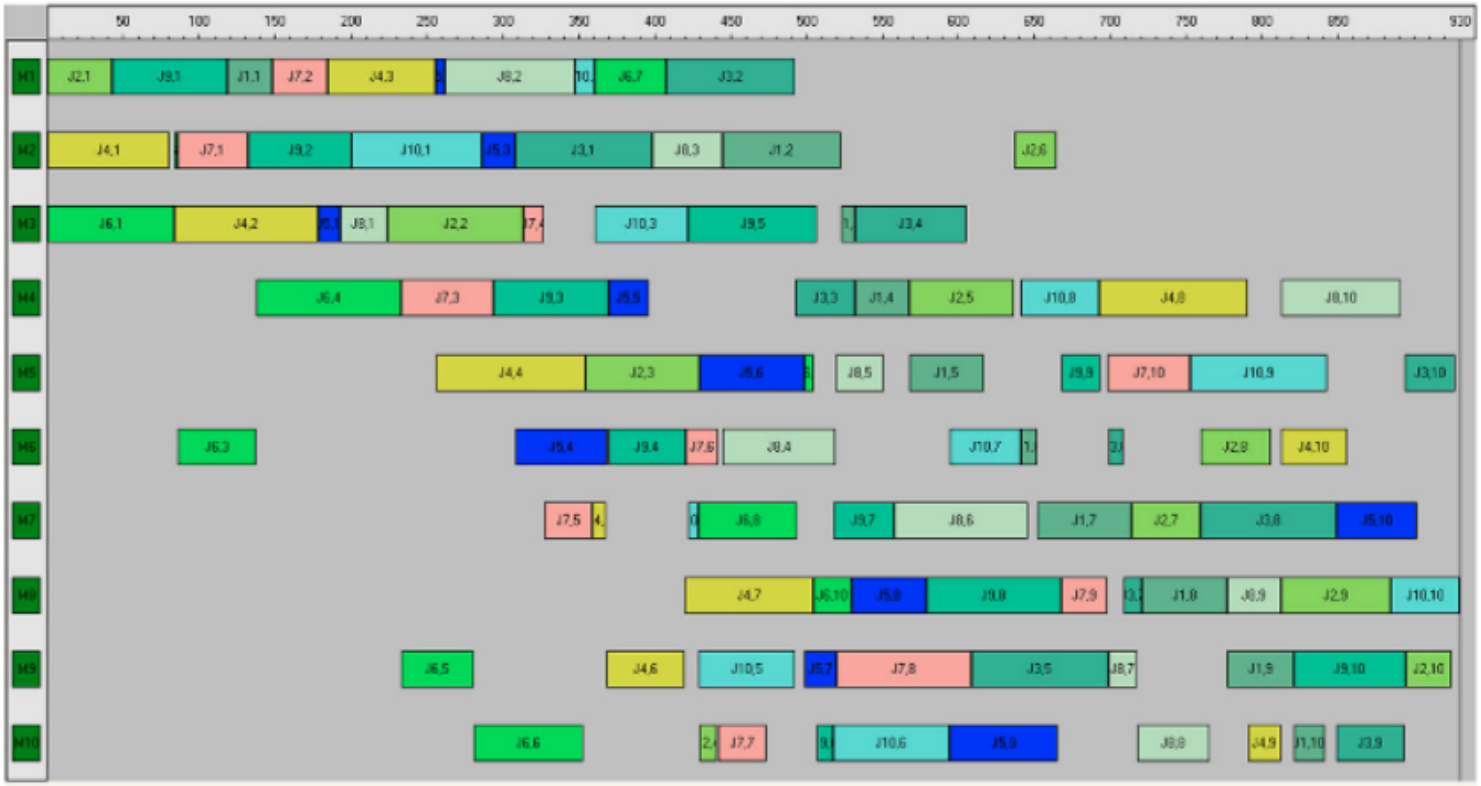
Conséquences sur la recherche en ordonnancement de 1980 à nos jours

- Développement du branch and bound "moderne" et de la **programmation par contraintes** pour le job-shop
- Invention des heuristiques de type **Large neighborhood search** (LNS) (*The shifting bottleneck procedure*)
- Développement des **recherches locales à haute performance** : voisinage des blocs
- Extensions au **job-shop complexe** et à l'ordonnancement de projet

subsection in toc subsection in toc subsection in toc

Branch and bound pour le job-shop : 10x10=930

[Carlier et Pinson, 1989, 1990] proposent un algorithme de branch and bound capable pour la première fois de résoudre une instance ouverte de $n = 10$ jobs et $n = 10$ machines.



solution optimale du FT10 (figure [Li 2011])

Schéma général des méthodes de *branch and bound* basées sur le graphe disjonctif

- Un nœud = une orientation partielle du graphe disjonctif
- A chaque nœud
  - Calculer les r_i et q_i des tâches par l'algorithme de Bellman
  - Calculer une borne inférieure du nœd LB
  - Tenter de mettre à jour la borne supérieure LB
  - Supprimer le nœud si $LB \geq UB$
  - Fixer des disjonctions supplémentaires (sélections immédiates) et ajuster les r_i et q_i
  - Brancher en orientant des disjonctions

Exemple de borne inférieure : $\max_{k=1,\ldots,m} h^*(\mathcal{J}_k)$

Exemple de borne supérieure : compléter les disjonctions non arbitrées par des heuristiques de liste.

Les sélections immédiates sur paire de disjonctions

Soit UB une borne supérieure (solution réalisable). Les sélections immédiates déduisent des arbitrages $i \prec j$ pour des paires $\{i, j\} \in D$.

$$r_i + p_i + p_j + q_j > UB \implies j \prec j \implies \begin{cases} q_j \leftarrow \max(q_j, q_i + p_i) \\ r_i \leftarrow \max(r_i, r_j + p_j) \end{cases}$$

Algorithme trivial en $\mathcal{O}(n^2)$ [Carlier (1982)], ein $O(n \log n)$ en utilisant un tas binaire [Carlier & Pinson 1994]

Les sélections immédiates sur ensembles ascendants

Soit UB une borne supérieure (solution réalisable). Déduction d'un arbitrage entre une tâche i un un ensemble J dit ascendant de i avec $i \notin J$ et $J \cup \{i\}$ une clique de D.

$$\begin{cases} h(j \downarrow J) = \min_{j \in J} r_j + \sum_{j \in J} p_j + p_i + \min_{j \in J} q_j > UB \\ h(j \prec J) = r_i + p_i + \sum_{j \in J} p_j + \min_{j \in J} q_j > UB \end{cases}$$

$$\Longrightarrow J \prec i \text{ c'est-à-dire } j \prec i, \forall j \in J$$

$$\Longrightarrow r_c \leftarrow \max \left(r_c, \max_{J' \subseteq J} (\min_{j \in J'} r_j + \sum_{j \in J'} p_j) \right)$$

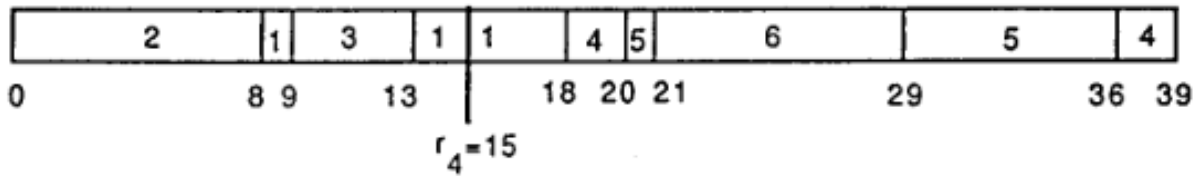
[Carlier & Pinson (1990)]

Les sélections immédiates sur ensembles ascendants

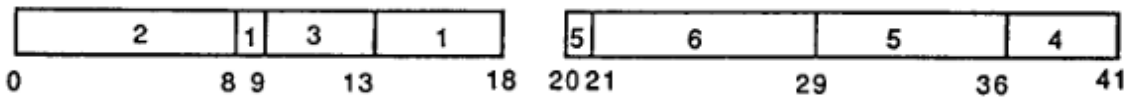
Un algorithme en $O(n^2)$ [Carlier & Pinson (1990)]

- pour chaque tâche $c \in C$ (C clique de disjonctions)
  - $J \leftarrow C \setminus \{c\}$; $K_c \leftarrow \{j \in J | q_j > q_c\}$
  - Ordonnancer les tâches avec JPPS : obtenir les dates de fin f_j, $j \in C$.
  - $K_c^- \subseteq K_c \leftarrow$ tâches terminées avant r_c
  - $K_c^+ \subseteq K_c \leftarrow$ tâches non terminées avant r_c de durées restantes p_j^+.
  - Chercher dans l'ordre croissant des q_j dans K_c^+ la première tâche j^* verifiant $r_c + p_c + \sum_{k \in K_c^+, q_j \geq q_{j^*}} p_j^+ + q_{j^*} > UB$
  - Si j^* existe, $K_c^* \leftarrow \{j \in K_c^+ | q_j \geq q_{j^*}\}$; Ajuster $r_c = \max_{j \in K_c^*} f_j$

i	1	2	3	4	5	6
r_i	4	0	9	15	20	21
p_i	6	8	4	5	8	8
q_i	20	25	30	9	14	16



$UB = 52$, $c = 4$, $K_c^+ = \{5, 6, 1\}$, avec $j^* = 5$, on obtient
$r_4 + p_4 + p_1^+ + p_5^+ + p_6^+ + q_5 = 39 + 14 = 53 > UB$ et $K_c^* = K_c^+$. On ajuste $r_4 = 36$.



Les sélections immédiates sur ensembles descendants

Déductions symétriques sur les q_i, également en $O(n^2)$ [Carlier & Pinson (1990)]

Le Edge-Finding et les autres algorithmes de filtrage de la contrainte disjonctive

L'algorithme de [Carlier & Pinson (1990)] est maintenant connu sous le nom de Edge-Finding.

Autre variante en $0(n^2)$ sans utiliser JPPS (voir TP Ronan Bocquillon dans cette école) : [Nuijten 94], [Martin & Shmoys 96]

Variante en $O(n \log n)$ [Carlier & Pinson, 94]

Autres algorithmes en $O(n \log n)$: Not First.Not Last, Detectable Peecedences [Vilim 2004] $\rightarrow$ CP Optimizer

Opérations globales (*shaving*) [Carlier & Pinson, 1994, Martin and Shmoys 1996] :
Chosir une tâche c et augmenter progressivement r_c en appliquant les ajustements à chaque fois. Si un échec arrive à $r_c = f_c$ alors on peut fixer $q_c = UB + 1 - (f_c + p_c)$.

Branchement [Carlier & Pinson 1994]

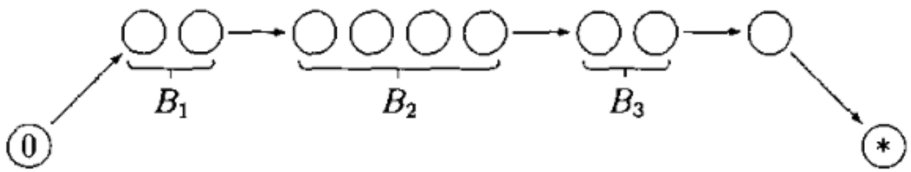
- Branchement basée sur la borne inférieure (JPS)
  - Trouver une tâche c et une clique J telle que (par exemple):

$$h(c \prec J) > UB, \quad h(c \downarrow J) \leq UB, \quad h(J \prec c) \leq UB$$

    alors générer deux nœuds :
    - c est ordonnancée après au moins une tâche de J
    - c est ordonnancée après toutes les tâches de J
  - brancher sur une paire de disjonction non arbitrée sinon

Branchement [Brucker et al, 1994]

Étant donné une solution réalisable représentée par une orientation complète π du graphe disjonctif, le makespan est donné par le chemin critique, $L(\pi)$ de longueur $l^*_{0,n+1}(\pi)$, décomposable en **blocs**.



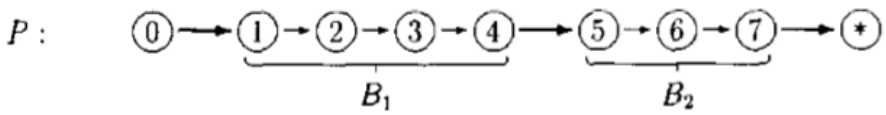
[Brucker et al. 1994]

Un bloc : un ensemble de tâches maximal du chemin critique consécutives sur la même machine.

Théorème

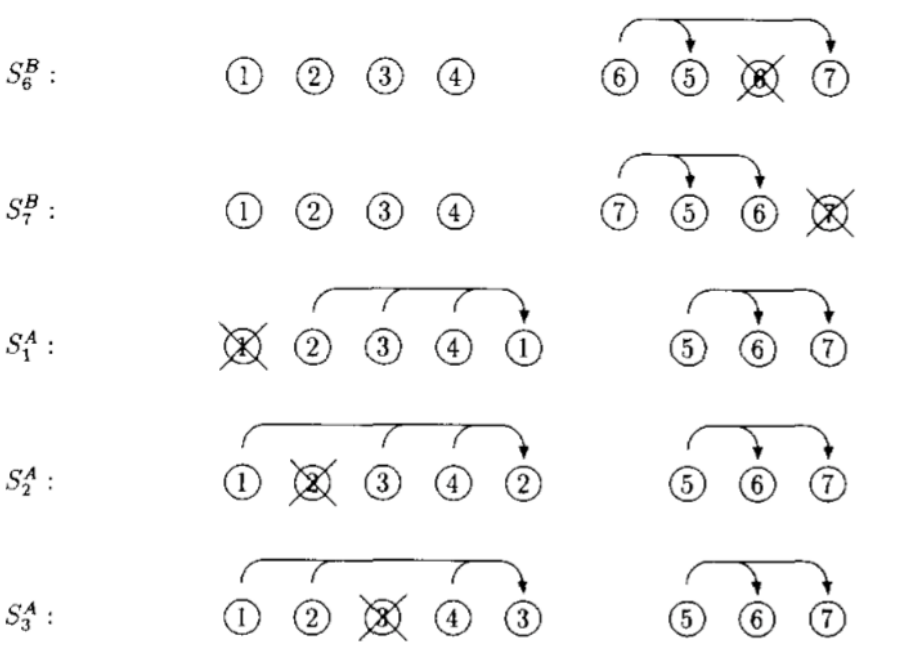
Pour qu'une orientation π' soit telle que $l^*_{0,n+1}(\pi') < l^*_{0,n+1}(\pi)$, alors au moins une tâche à l'intéreur d'un bloc de $L(\pi)$ doit être ordonnancée soit avant la première soit après la dernière tâche du bloc

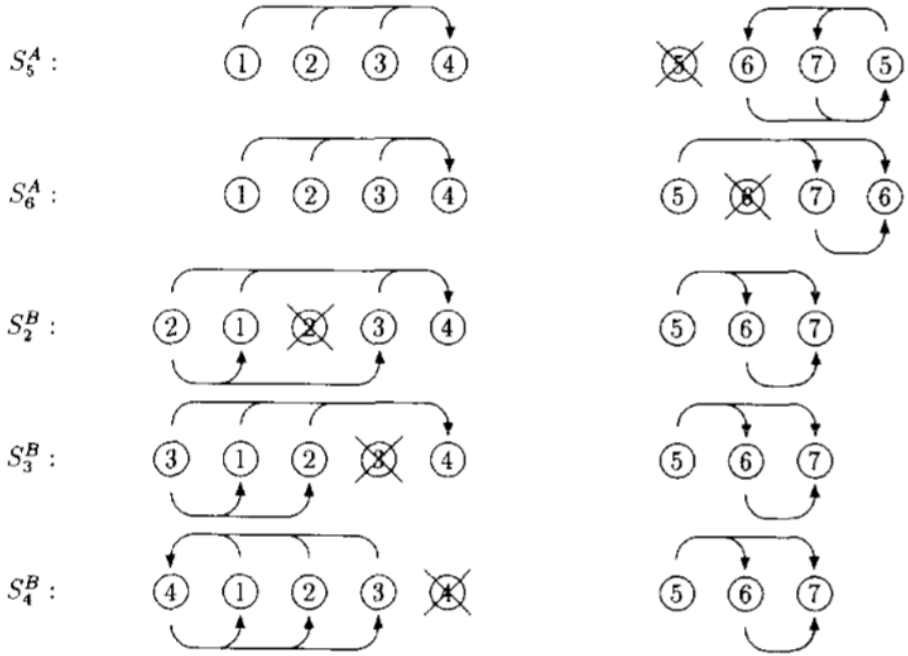
Branchement [Brucker et al, 1994]



notation : S_i^B (S_i^A) la tâche i est ordonnancée avanr (après) son bloc.

Les S_i^X, pour les tâches d'un bloc et un $X \in \{B, A\}$ donné sont explorés consécutivement. Par exemple après l'exploration des sous-arbres S_i^B d'un bloc commençant par j on peut fixer $j \prec i$ pour tout $i \neq j$ du bloc.





Brand and bound de [Grimes and Hébrard 2015]

Approche de programmation par contraintes *Light Weight*

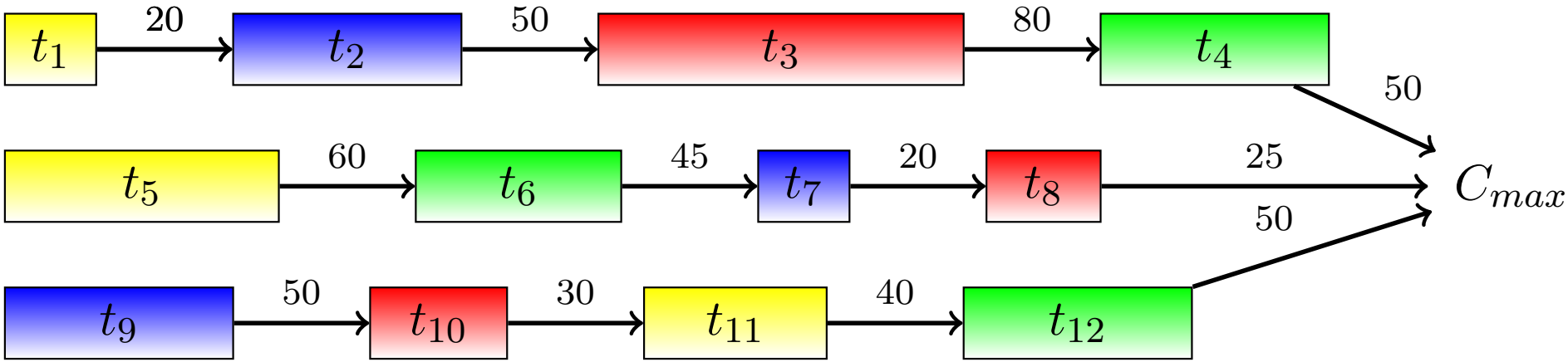
Calculs légers à chaque nœud, branchement sur une unique disjonction

Choix de la disjonction à arbitrer selon une pondération évoluant en fonction des échecs rencontrés lors des précédents branchements sur cette disjonction

Choix du sens de l'arbitrage en ce guidant sur la meilleure solution trouvée

Utilisation de redémarrages géométriques, et de *no-goods* à chaque redémarrage.

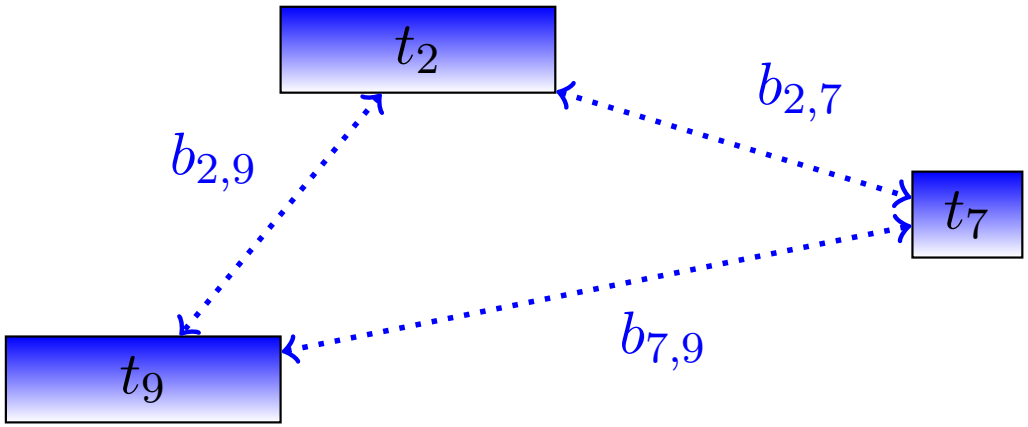
Modèle de programmation par contraintes *Light Weight*



Model

- A Variable for the start time of each task: $t_i \in [0, \ldots, C_{max}]$.
  - Precedence constraints: $t_i + p_i \leq t_{i+1}$.

Modèle de programmation par contraintes *Light Weight*



Model

- A Variable for the start time of each task: $t_i \in [0, \ldots, C_{max}]$.
  - Precedence constraints: $t_i + p_i \leq t_{i+1}$.
- A Boolean Variable standing for the relative order of each pair of conflicting tasks (disjunct):
  - Binary Disjunctive constraints: $b_{ij} = \begin{cases} 0 \Leftrightarrow t_i + p_i \leq t_j \\ 1 \Leftrightarrow t_j + p_j \leq t_i \end{cases}$

Propagation

Assurer la cohérence de borne :

1. sur chaque contraintes de précédence $(i,j) \in E$:
   $\min(t_i) + p_i < \min(t_j) \implies \min(t_j) \leftarrow \min(t_i) + p_i$ et
   $\max(t_j) - p_i < \max(t_j) \implies \max(t_i) \leftarrow \min(t_j) - p_i$
2. sur chaque contrainte disjonctive $\{i,j\} \in D$
   - Si $b_{ij} = 0$ ou 1, cohérence de borne de la contrainte de précédence
   - $\min(t_i) + p_i > \max(t_j) \implies b_{ij} \leftarrow 0$ et
     $\min(t_j) + p_j > \max(t_i) \implies b_{ij} \leftarrow 1$

Complexité : processus de réveil des en contraintes dans le pire des cas en $O(UB * (|D| + |E|))$ (rarement atteint en pratique) Equivalence :

1. au maintient des r_i et q_i des tâches dans le graphe disjonctif partiellement orienté :
   $r_i = \min(t_i) \leftarrow l^*(0,i)$ et $q_i = UB - \max(t_i) - p_i \leftarrow l^*(i,n+1)$
2. Aux sélections immédiates sur les paires de disjonction

Complexité : application jusqu'à atteindre un point fixe, même complexité si implémentation naïve mais [Peridy et Rivreau 2005] proposent une procedure stable pour tous les ajustements en $O(mn^2)$.

Branchement dirigé par les échecs

Choix de la variable b_{ij} pour l'heuristique de *domain/weighted degree* [Boussemart et al 2004]

- Calcul d'un poids $w(i,j)$ pour chaque disjonction : nombre de fois ou la propagation de la disjonction a provoqué un échec:

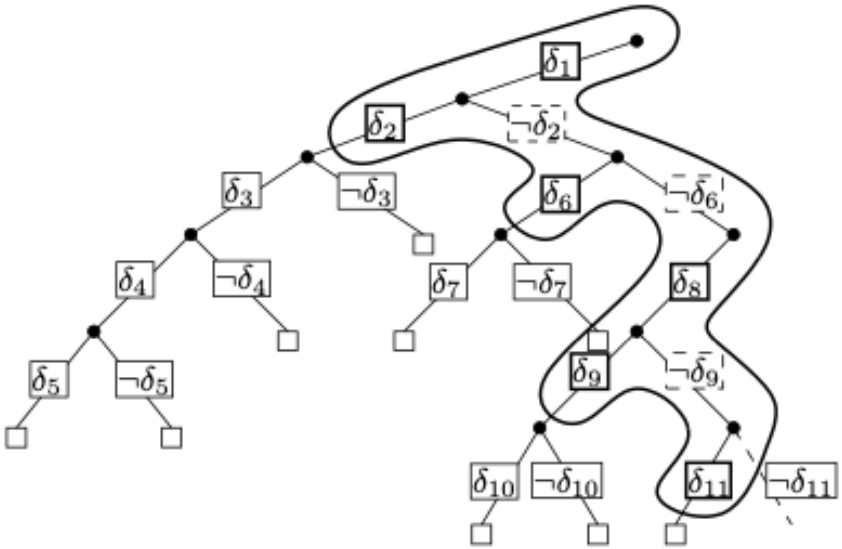
- Deux heuristiques différentes : choix de la variable b_{ij} qui minimise $\frac{dom(t_i)+dom(t_j)}{w(i,j)}$ ou $\frac{dom(t_i)+dom(t_j)}{w(t_i)+w(t_j)}$ avec $domt_i = UB - q_i - p_i - r_i$ et $w(t_i) = \sum_{\{i,j\} \in D} w(i,j)$.

Choix de la valeur : mémoriser la meilleure solution trouvée (UB) et brancher sur la valeur de la variable $b_{i,j}$ dans cette solution.

Redémarrage avec *no-goods*

Redémarrage à structure géométrique : on fixe un nombre d'échecs s et un facteur multiplicatif r On redémarre la recherche selon lorsque le nombre d'échecs atteint sr puis sr^2 puis sr^3, etc....

Lorsqu'on arrête la recherche, on mémorise un *no-good* par branche "de droite" (2 ème choix de valeur) car la branche de gauche a été totalement explorée.



Exemple : $\delta_1 : b_{1,2} = 1$, $\delta_2 : b_{3,6} = 0$,
$\delta_8 : b_{1,3} = 1$, $\delta_9 : b_{2,3} = 0$, $\delta_{11} : b_{1,7=0}$

no goods $\Delta_1 = \{\delta_1, \neg\delta_2, \neg\delta_6, \delta_8, \neg\delta_9, \delta_{11}\}$,
$\Delta_2 = \{\delta_1, \neg\delta_2, \neg\delta_6, \delta_8, \delta_9\}$,
$\Delta_3 = \{\delta_1, \neg\delta_2, \delta_6\}$, $\Delta_4 = \{\delta_1, \delta_2\}$.

Résolution
$(\neg\delta_1 \vee \delta_2 \vee \neg\delta_6) \wedge (\neg\delta_1 \vee \neg\delta_2) \vdash \neg\delta_1 \vee \neg\delta_6$

no goods réduits par résolution
$\Delta_1 = \{\delta_1, \delta_8, \delta_{11}\}$, $\Delta_2 = \{\delta_1, \delta_8, \delta_9\}$,
$\Delta_3 = \{\delta_1, \delta_6\}$, $\Delta_4 = \{\delta_1, \delta_2\}$.

[Lecoutre et al 2007]

Exploitation *no-goods*

Les no-goods sont mémorisés au sein d'une contrainte globale

.

La contrainte est propagée lorsqu'un b_{ij} est fixé pour appliquer si possible la propagation unitaire (*UP*) :

Exemple : no-good $\Delta_1 = \{b_{1,2} = 1, b_{1,3} = 1, b_{1,7} = 0\}$ (i.e. clause $\neg b_{1,2} \vee \neg b_{1,3} \vee b_{1,7}$)
On fixe $b_{1,2} = 1 \implies$ pas de propagation
On fixe $b_{1,7} = 0 \implies b_{1,3} = 0$ (unit propagation)

Remarque : Pour plus d'efficacité dans la propagation unitaire, on utilise la technique des *watched literals* : on a un pointeur sur deux littéraux (variables) de chaque clause et on associe à chaque variable la liste des clauses dans laquelle elle intervient. Si la variable fixée n'est pas un watched littéral de la clause, il n'y a rien à faire. Sinon on cherche un autre watched litteral et s'il n'y en a pas : UP !

Comparaison expérimentale sur le problème de Job-Shop

Instances de Job-Shop de 6 x 6 à 50 x 20 instances

Comparaison du Branch and Bound [Grimes & Hébrard 2015] (LW) avec CP Optimizer 12.5 (CPO)

Mesures de performance (10 runs par instance):

- APRD : $\frac{C_{max}(Alg)-C^*_{max}}{C^*_{max}} * 100$ où C^*_{max} est le meilleur makespan trouvé par les deux algorithmes.
- Opt (atleast1) : nombre de fois où au moins une exécution a prouvé l'optimalité
- Opt (every) : nombre de fois où au moins toutes les exécutions ont prouvé l'optimalité

Résultats:

#inst.	APRD				Opt			
	LW		*CPO*		*LW*		*CPO*	
	Best	Avg	Best	Avg	atleast1	every	atleast1	every
152	1.14	1.81	**0.08**	0.54	**67**	63	66	58

Remarque : CPO est meilleur sur les grandes instances.

subsection in toc subsection in toc subsection in toc

Prémisse des recherche locales à voisinage étendu (LNS)

The shifting bottleneck heuristic [Adams et al 1988]

Partir d'un graphe disjonctif partiellement orienté (une partie des machines (M) est totalement séquencée, l'autre (M') non).

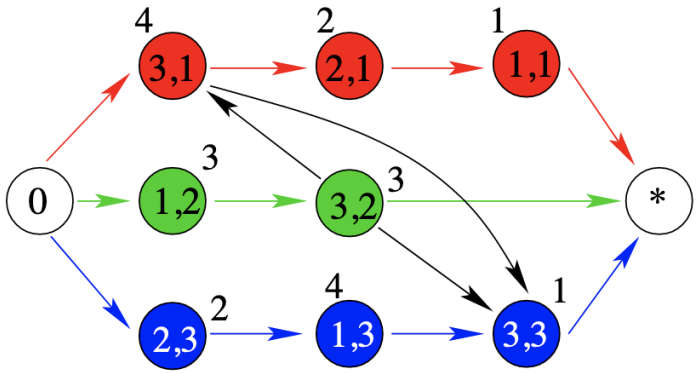
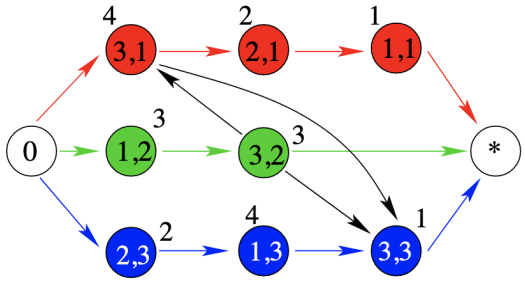


Fig. [Hurink 2005]

$M = \{M_3\}$, $C_{max}(M) = 13$

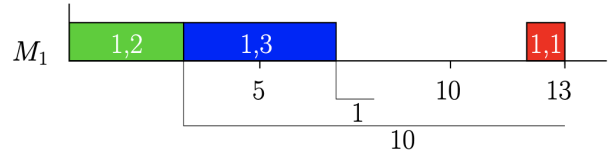
The shifting bottleneck heuristic [Adams et al 1988]

Etape 1. Recherche la machine non-séquencée goulet m^* (*bottleneck*) en résolvant indépendamment chaque machine par l'algorithme de [Carlier 1982].



Machine M_1:

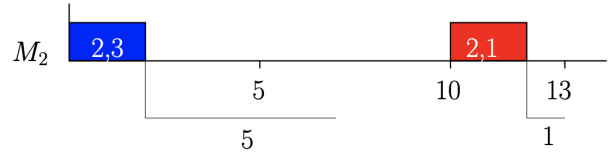
(i,j)	$(1,1)$	$(1,2)$	$(1,3)$
r_{ij}	12	0	2
q_{ij}	0	10	1
p_{ij}	1	3	4



$f(M_1) = 13$

Machine M_2:

(i,j)	$(2,1)$	$(2,3)$
r_{ij}	10	0
q_{ij}	1	5
p_{ij}	2	2



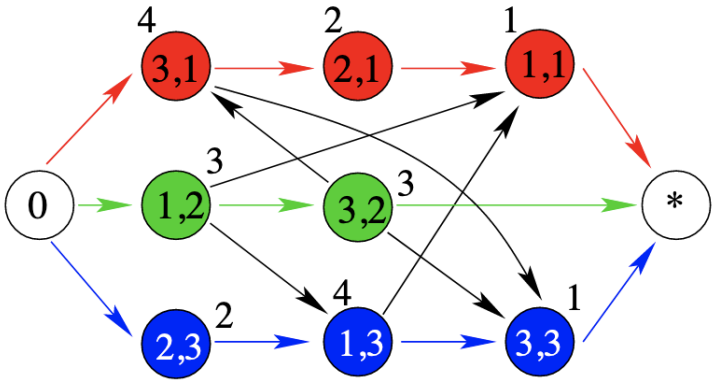
$f(M_2) = 13$

Fig. [Hurink 2005]

La machine goulet est m^* =M1

The shifting bottleneck heuristic [Adams et al 1988]

Etape 2. Fixer la séquence de la machine goulet m^* en orientant les arcs disjonctifs correspondant et l'ajouter à M



$M = \{M1, M3\}$, $C_{max}(M) = 13$.

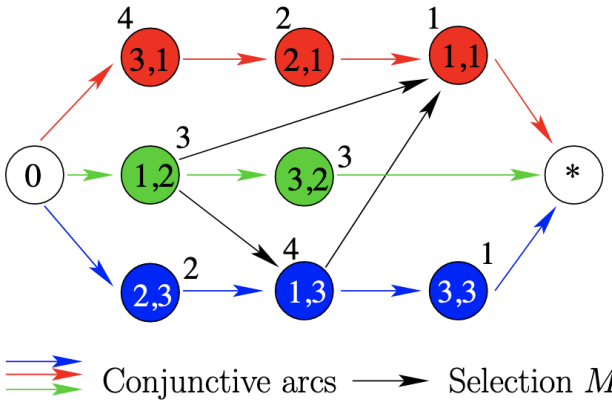
The shifting bottleneck heuristic [Adams et al 1988]

Etape 3. Pour chaque machine séquencée $m \in M \setminus \{m^*\}$ (phase de réoptimisation locale)

- supprimer (seulement) les arcs de m et recalculer les r_i et q_i
- ordonnancer m par l'algorithme de Carlier et fixer la séquence de m en ajoutant les arcs trouvés.

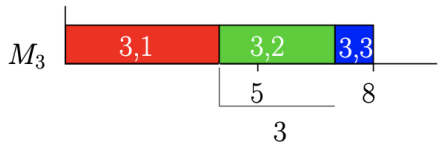
Revenir à l'étape 1 si $|M| \neq m$

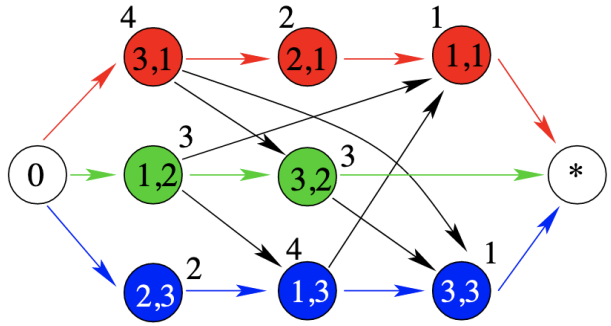
rescheduling Machine M_3



$C_{\max} = 8,\ f(M_3) = 0$

(i,j)	$(3,1)$	$(3,2)$	$(3,3)$
r_{ij}	0	3	7
q_{ij}	3	0	0
p_{ij}	4	3	1





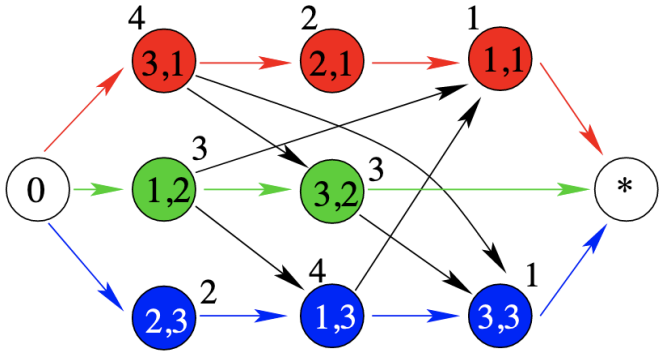
$C_{\max}(M) = 8$

The shifting bottleneck heuristic [Adams et al 1988]

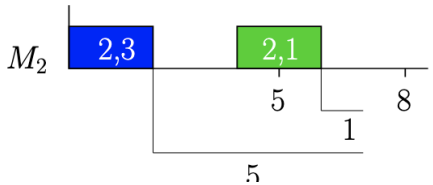
Etape 1. Recherche la machine non-séquencée goulet m^* (*bottleneck*) en résolvant indépendamment chaque machine par l'algorithme de [Carlier 1982].

$M = \{M1, M_3\}$ donc $m^* = M_2$ est le bottleneck

- graph G:



(i,j)	$(2,1)$	$(2,3)$
r_{ij}	4	0
q_{ij}	1	5
p_{ij}	2	2

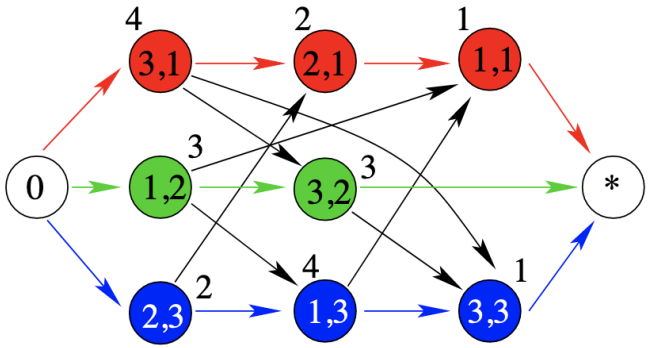


$f(M_2) = 7$

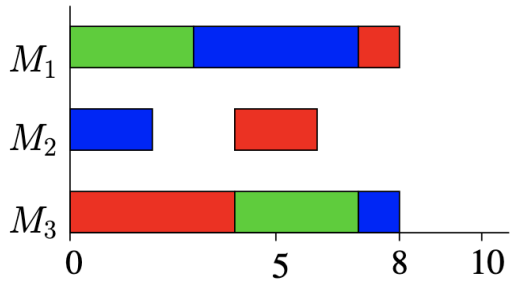
The shifting bottleneck heuristic [Adams et al 1988]

Etape 2. Fixer la séquence de la machine goulet m^* en orientant les arcs disjonctifs correspondant et l'ajouter à M

- Graph G:



- Corresponding Schedule:

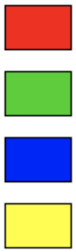


$M = \{M1, M2, M3\}$, $C_{max}(M) = 8$.

Il y a un bug dans SB...

Problème de départ

Jobs:



$(1,1) \rightarrow (2,1)$

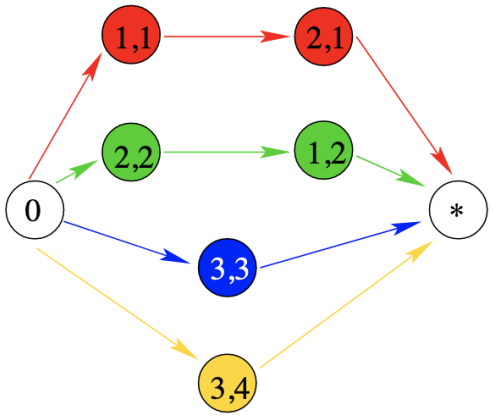
$(2,2) \rightarrow (1,2)$

$(3,3)$

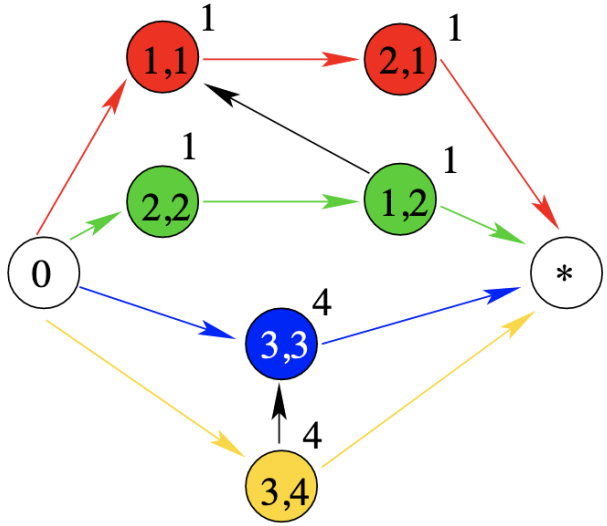
$(3,4)$

Processing Times: $p_{11} = 1,\ p_{21} = 1$
$p_{22} = 1,\ p_{12} = 1$
$p_{33} = 4$
$p_{34} = 4$

Initial graph G:



après 2 itérations



$M = \{M_3, M_1\},\ C_{\max}(M) = 8$

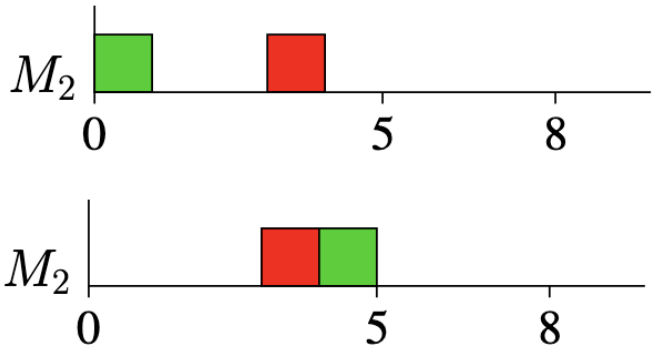
.... des solutions irréalisables peuvent être produites !

troisième itération $m^* = M_2$.

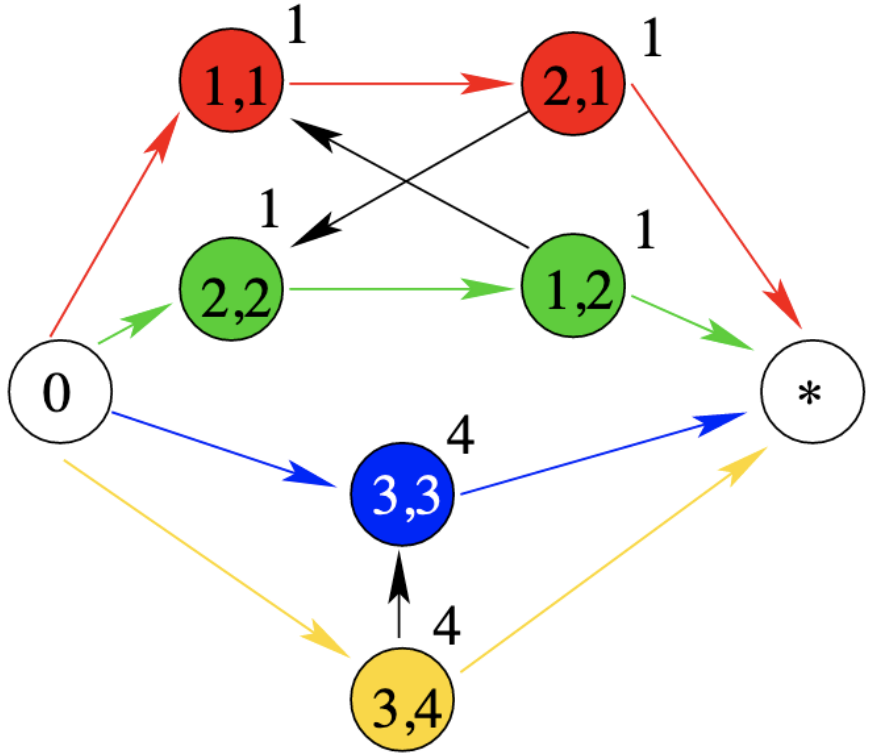
- 3. iteration: only M_2 unscheduled

(i,j)	p_{ij}	r_{ij}	q_{ij}
$(2,1)$	1	3	0
$(2,2)$	1	0	3

- Possible schedules for M_2:



si on chosit le deuxième ordonnancement :



il y a un circuit. Il y avait une précédence entre $(2,2)$ et $(2,1)$ non capturée par les r_i et q_i

[Dauzère-Pérès & Lasserre 93] et [Balas et al 1995] modifient l'algorithme de Carlier pour traiter ce cas.

Recherche locale basée sur le graphe disjonctif

L'algorithme de shifting bottleneck définit une recherche locale à (très) grand voisinage, et le passage d'une solution à une autre est très couteux.

L'utilisation de voisinage de taille plus petite permet de générer plus de voisins en un temps plus réduit.

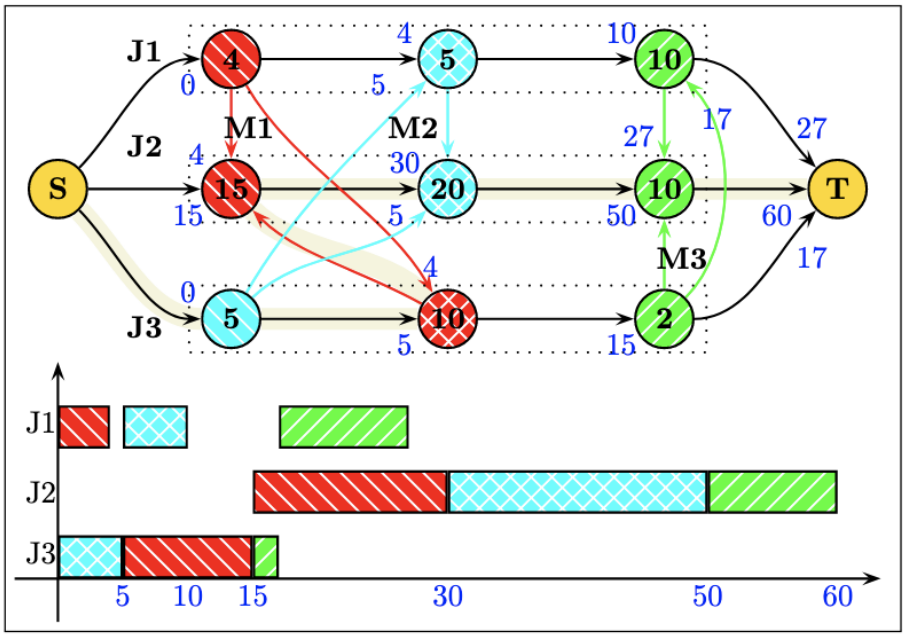
Les voisinages basés sur l'analyse du chemin critique du graphe disjonctif ont été très étudiés.

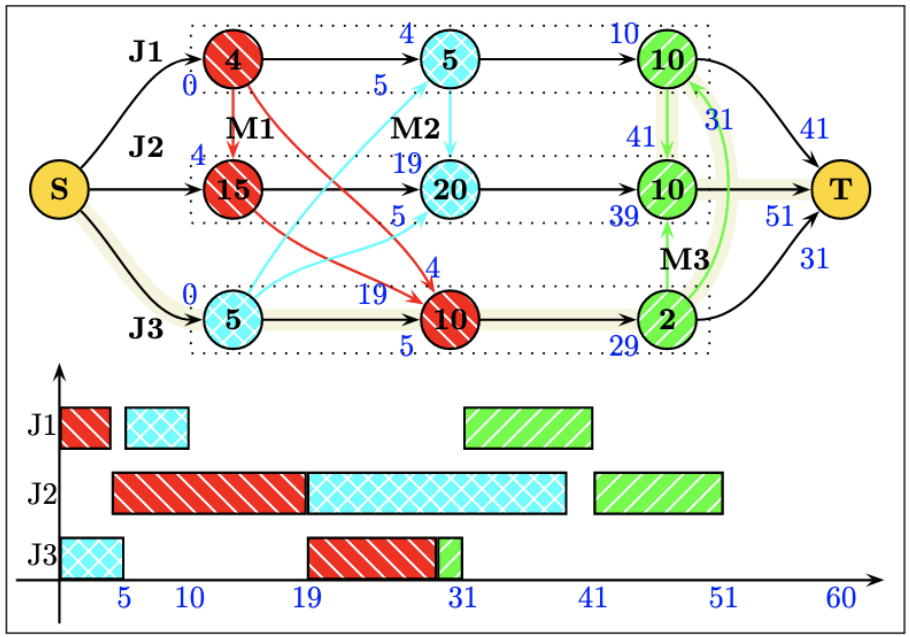
Théorème

Pour améliorer une borne supérieure $UB = l^*(0, n+1)$ obtenue par une orientation de plus long chemin $P^*(0, n+1)$ il faut inverser au moins un arc de $P^*(0, n+1)$

Recherche locale basée sur le graphe disjonctif

Exemple d'inversion d'un arc sur le chemin critique





[Binato et al 2001]

Recherche locale basée sur le graphe disjonctif

Voisinages d'échange basés sur les blocks

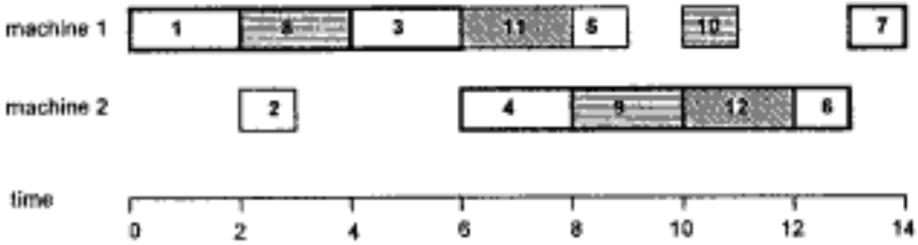


fig [Nowicki, Smutnicki 96]

- Voisinage de [van Laarhoven et al 1992] :
  inverser un arc $(u, v) \in P^*(0, n)$ de deux tâches consécutives dans un bloc
- Voisinage de [Nowicki & Smutnicki 1996] :
  inverser un arc $(u, v) \in P^*(0, n)$ de deux tâches consécutives aux extrémités d'un bloc

Théorème

Un tel échange est toujours réalisable. Un échange de deux tâches internes ne peut améliorer le makespan

Recherche locale basée sur le graphe disjonctif

Voisinage de [Balas et Vazacopoulos 1998] :

Prendre un arc $(u, v) \in P^*(0, n)$ de tâches non nécessairement consécutives

Inverser (u, v) de telle sorte que

- u soit inséré immédiatement après v (forward) ou
- v soit inséré immédiatement avant u (backward)

Réalisabilité et impact sur le makespan en $O(n)$ mais conditions suffisantes.

Ex. conditions suffisantes de réalisabilité :
Soit $\Gamma^-(u)$ ($\Gamma^+(u)$) le prédécesseur (successeur) de u dans le job.

Théorème

Si ni $\Gamma^-(u)$ ni $\Gamma^+(v)$ ne sont sur le chemin critique alors échanger u et v ne peut réduire le makedspan.
Si $\Gamma^+(v)$ est sur le chemin critique et $l^*(v, n+1) \geq l^*(\Gamma^+(u), n+1)$ alors l'échange forward est réalisable ;
Si $\Gamma^-(u)$ est sur le chemin critique et $l^*(0, u) + p_u \geq l^*(0, \Gamma^-(v)) + p_{\Gamma^-(v)}$ alors l'échange backward est réalisable ;

Intégration dans des Metaheuristiques et résultats

Les voisinages sont utilisés par diverses metaheuristiques (Recherche tabou, recuit simulé, arbres de voisinage...) avec différents ingrédients d'intensification et de diversification (path relinking, ...).

Comparaisons d'une méthode Tabou très performance [Peng et al 2015] avec le branch-and-bound LW [Grimes & Hébrard 2015] (mesure APRD).

Problem	#inst.	*LW*		*CPO*		SOA Systematic		SOA Heuristic	
		Best	Avg	Best	Avg	Best	Ref	Best	Ref
JSP	115	1.49	2.18	0.48	1.01	0.81	Beck (2007)	**0.01**	Peng et al. (2014)

subsection in toc subsection in toc subsection in toc

Et en pratique ?

Le modèle de Job-shop ne modélise pas fidèlement les situations pratiques....

- Ressources flexibles
- temps de préparation dépendant de la séquence
- Tâches multi-ressources
- Calendriers de disponibilités des ressources
- Machines Batch
- Précédences généralisées (*time lags*)
- Ressources cumulatives
- Possibilité de Préemption, découpage de lot
- ....

Si le modèle de graphe disjonctif s'adapte bien à ces situations....
.... il n'en va pas de même de tous les algorithmes

Job-shop flexible (FJSP)

FJSP

Job i	Number of operations n_i	Operation v	Eligible machines
1	4	1	{1}
		2	{1;2}
		3	{2;3}
		4	{3}
2	2	5	{1;3}
		6	{2}
3	3	7	{3}
		8	{1;3}
		9	{2}

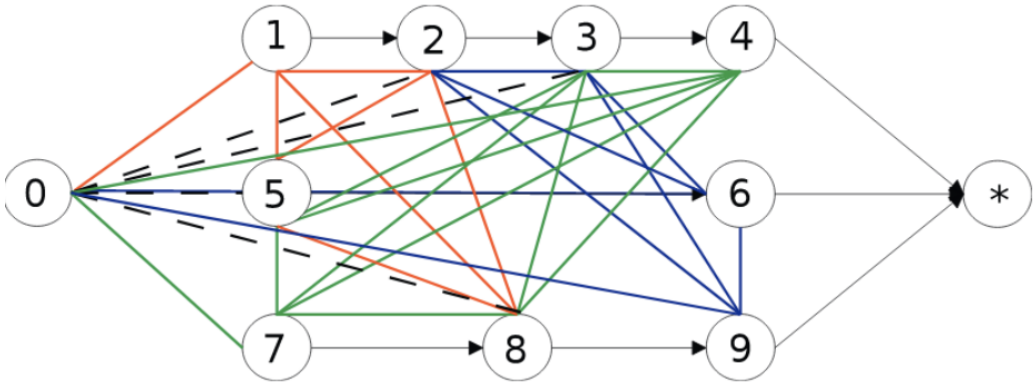
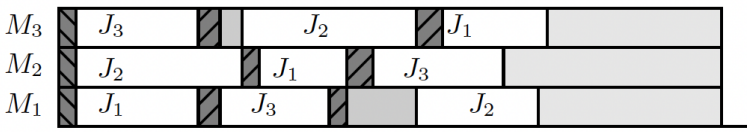
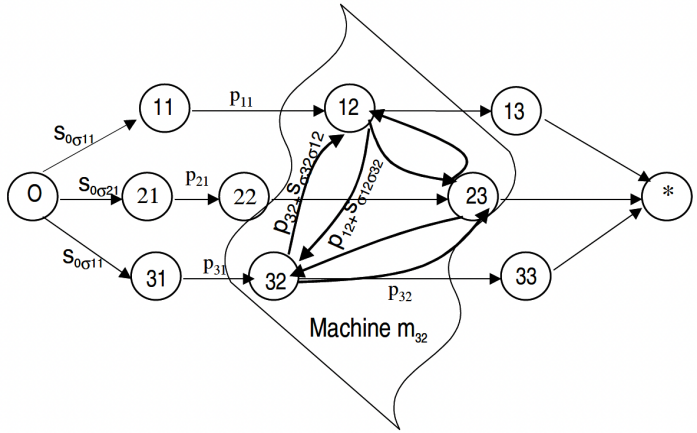


Fig [Shen et al 2018]

- Après affectation à une machine, les arcs inutiles de D sont supprimés
- Augmentation considérable de la combinatoire
  [Dauzère-Pérès et Paulli 1997] proposent des conditions suffisantes de réalisabilité d'un movement de déplacement d'une tâche u entre deux tâches v et w: $r_v < r_{\Gamma^+(u)} + p_{\Gamma^+(u)}$ et $r_w + p_w > r_{\Gamma^-(u)}$

Job-shop avec temps de préparation dépendant de la séquence (SDST-JSP)





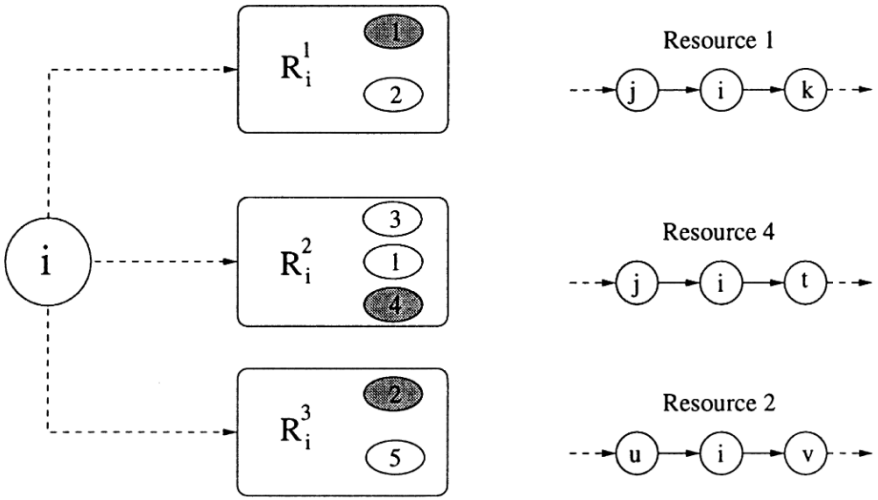
- Règle des blocs de [Nowicki & Smutnicki 96] et Conditions suffisantes de mouvement plus valables
- [Shen et al 2018] proposent for le F-SDST-JSP une condition pour déplacer u entre v et w :

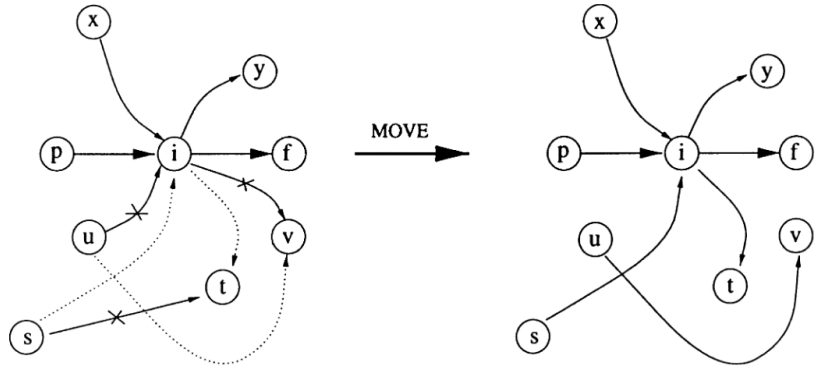
$$\max(r_x, r_{\Gamma-(v)}) < r_{\Gamma+(u)} + p_{\Gamma+(u)} \text{ pour } (x, v) \in \Pi \text{ et}$$

$$\min(r_y + p_y, r_{\Gamma+(w)} + p_{\Gamma+(w)}) > r_{\Gamma-(v)} \text{ pour } (w, y) \in \Pi$$

- Branch and bound [A. et Feillet 2008] utilisent des relaxations de TSPTW, et l'intégration des setup times dans le Edge Finding.

Job-shop multi-ressources (MR-JSP)





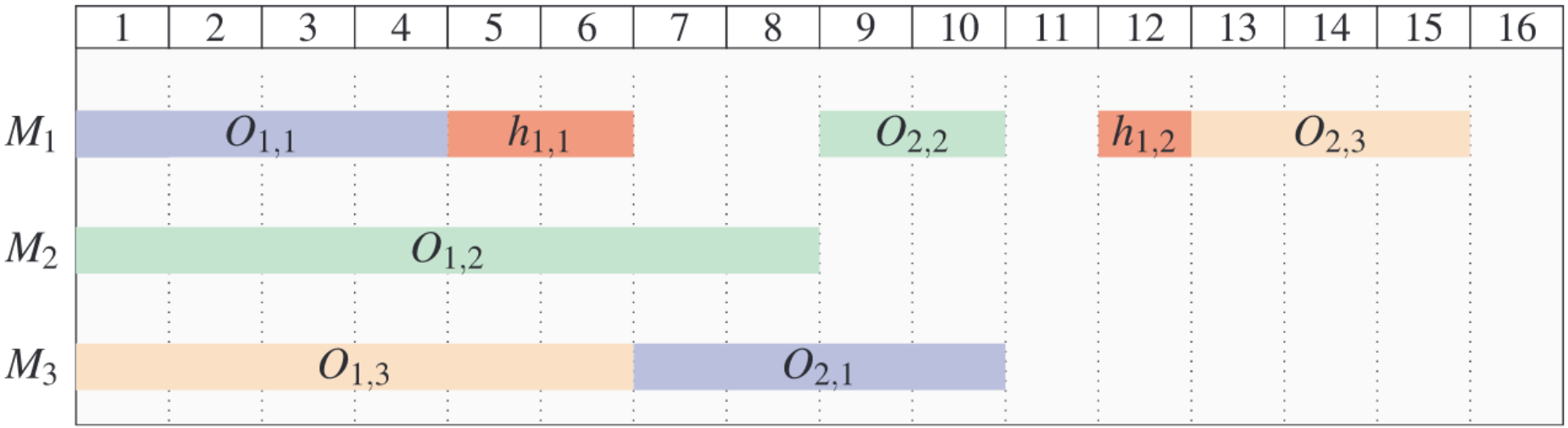
$\mathcal{R}_i^k$ ensemble des machines possibles pour la $k^{\text{ème}}$ machine. $\mathcal{F}(i)$ successeurs i (ressource et travail) $\mathcal{P}(i)$ prédécesseurs of i (ressource et travail)

- Conditions plus difficiles pur déplacer une tâche.
- [Roux et al. 1998] montrent que i peut être déplacée sur sa $k^{\text{ème}}$ machine entre s et t, successeurs sur $l \in \mathcal{R}_i^k$ si

$$r_s < \min_{f \in \mathcal{F}(\rangle) \setminus \{v\}} (r_f + p_f) \text{ and } r_t + p_t > \max_{p \in \mathcal{P}(\rangle) - \{u\}} r_p$$

où u et v sont le prédecesseur et le successeur actuel sur la $k^{\text{ème}}$ machine.

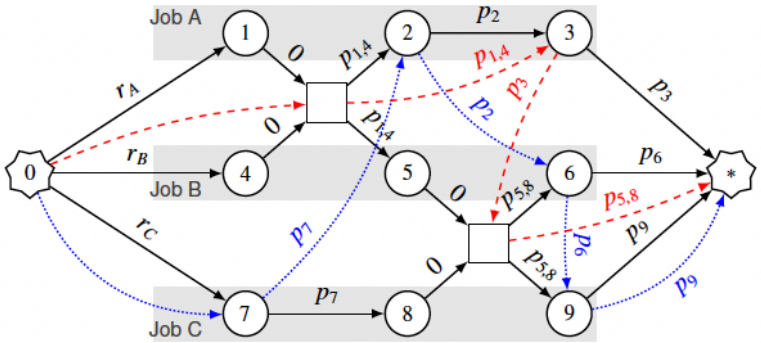
Job-shop avec calendrier de disponibilité des ressources



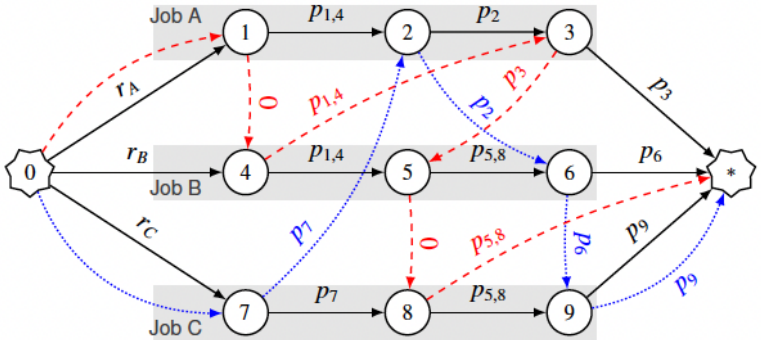
- nécessité d'adapter les calculs de plus longs chemins
- problème de déterminaiton de criticalité
- définition de voisinages dans [Tamssaouet et al 2018] pour estimer le cout d'échange de deux tâches.

Job-shop avec machines batch

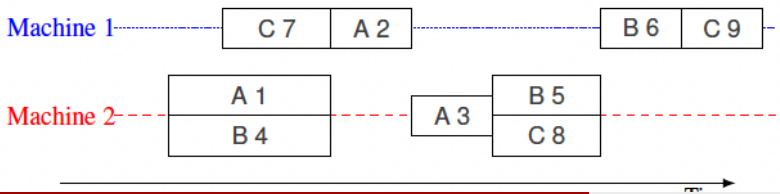
[Knopp et al 2017] Semiconductor manufacturing



(a) *Batch-Aware Conjunctive Graph*



(b) *Batch-Oblivious Conjunctive Graph*

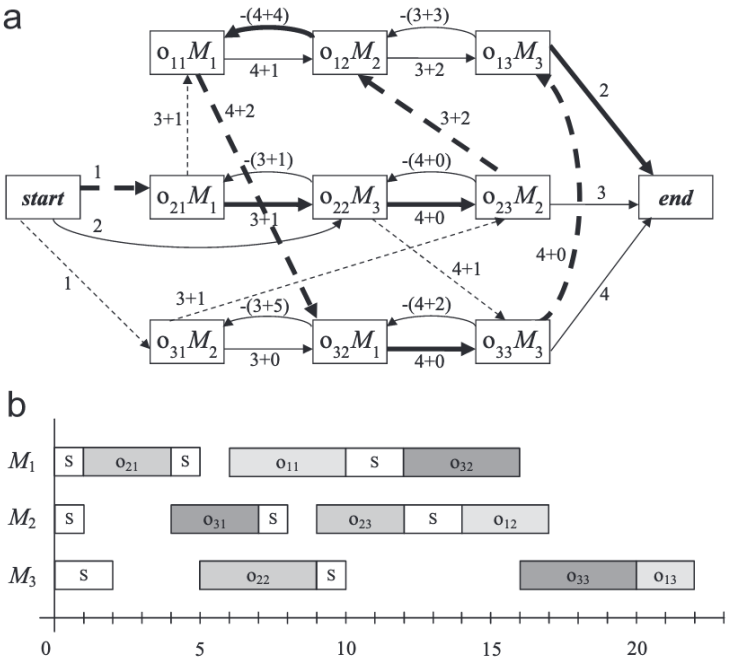


La représentation des batches nécessite des nœuds additionnels qui dépendent des décision

L'approche "Batch-oblivious" permet d'éviter cette modification en changeant le poids de l'arc entre un sommet u et son successeur w en assurant l'invariant suivant

si $l_{u,w} = 0$ alors
$S_u \geq S_{\Gamma^-(w)} + p_{\Gamma^-(w)}$
sinon $l_{u,w} = p_u$

Job-shop avec time lags



- Difficulté d'obtenir des solutions voisines réalisables
- [Gonzalez et al 2015] proposent deux structures de voisinage : une qui relâche les contraintes de time lags (swap classique) et l'autre qui cherche à retrouver la faisabilité lorsque les time lags sont violés.

Comparaisons expérimentales

Comparaisons des méthodes de l'état de l'art avec le branch-and-bound LW [Grimes & Hébrard 2015] (mesure APRD).

Table 2 APRD comparison with the State of the Art

Problem	#inst.	*LW*		*CPO*		SOA Systematic		SOA Heuristic	
		Best	Avg	Best	Avg	Best	Ref	Best	Ref
OSP	175	**0**	**0**	**0**	0.01	**0**	Malapert et al. (2008)	0.04	Sha and Hsu (2008)
JSP	115	1.49	2.18	0.48	1.01	0.81	Beck (2007)	**0.01**	Peng et al. (2014)
ET-JSP	42	**3.85**	16.14	6.27	34.15	274.4	Danna and Perron (2003)	100.46	Danna and Perron (2003)
SDS-JSP	15	**0.04**	0.56	0.19	0.52	2.06	Artigues and Feillet (2008)	0.14	González et al. (2008)
TL-JSP	27	**0**	0.04	0.06	0.08	6.00	Artigues et al. (2011)	15.60	Caumond et al. (2008)
NW-JSP	52	2.16	4.47	2.74	4.41	-	van den Broek (2009)	**0.18**	Bürgy and Gröflin (2012)

La méthode LW est très compétitive sur les variantes (TL, SDST) JSP.

Un problème industriel avec multi-ressources flexibles, time lags, calendriers

subsection in toc subsection in toc subsection in toc

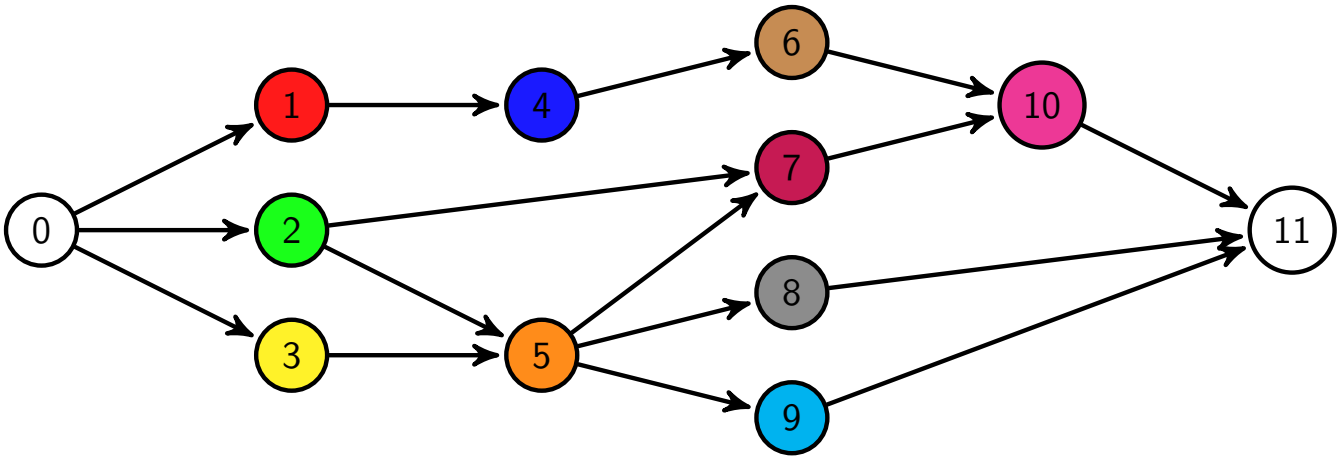
Le problème d'ordonnancement de projet à moyens limités

- Un problème d'optimisation combinatoire situé au cœur de nombreuses applications
  - Gestion de projet, gestion de production, génie des procédés, architectures de processeurs, ...
- Le RCPSP: un problème NP-difficile qui pose un défi computationnel depuis les années 60
  - Jeux de données: PAT [Patterson 1984], ALV [Alvarez-Valdes and Tamarit 1989], KSD [Kolisch, Sprecher and Drexl 1995,1997] (**PSPLIB**), BL [Baptiste and Le Pape 2000], PACK [Carlier and Néron 2003].
  - 48 (sur 480) instances à 60 activités et 4 ressources encore ouvertes de la PSPLIB

```
25      4       108             Fri Jul  6 18:49:10 2001  V. Valls, M. Quintanilla, F. Ballestin
25      5       98              Wed Feb 20 10:39:57 2002  M. Palpant, C. Artigues, P. Michelon
25      6       112             Fri May 30 10:21:35 2003  A. Stolyar, Yu. Kochetov
25      7       90              Wed Feb 20 10:39:59 2002  M. Palpant, C. Artigues, P. Michelon
```

RCPSP: données du problème

- R ensemble de ressources, disponibilité limitée $B_k \geq 0, k \in R$,
- A ensemble d'activités (tâches), durée $p_i \geq 0, i \in A$, demande $b_{ik} \geq 0$ pour $k \in R$,
- E ensemble de contraintes de précédence (i, j), $i, j \in A$, $i < j$
- $\mathcal{T} = [0, T]$ intervalle de temps (horizon d'ordonnancement)



$|R| = 1, B = 4, \mathcal{T} = [0, 30]$

i	p_i	b_i
1	3	2
2	5	3
3	1	3
4	3	1
5	2	1
6	4	2
7	5	3
8	6	1
9	4	1
10	4	1

RCPSP: variables, objectif et contraintes

- $S_i \geq 0$ date de début de l'activité i
- $C_{\max} = S_{n+1}$ durée totale du projet

RCPSP (formulation conceptuelle)

$$\min_{S \in \mathcal{S}_T} S_{n+1}$$

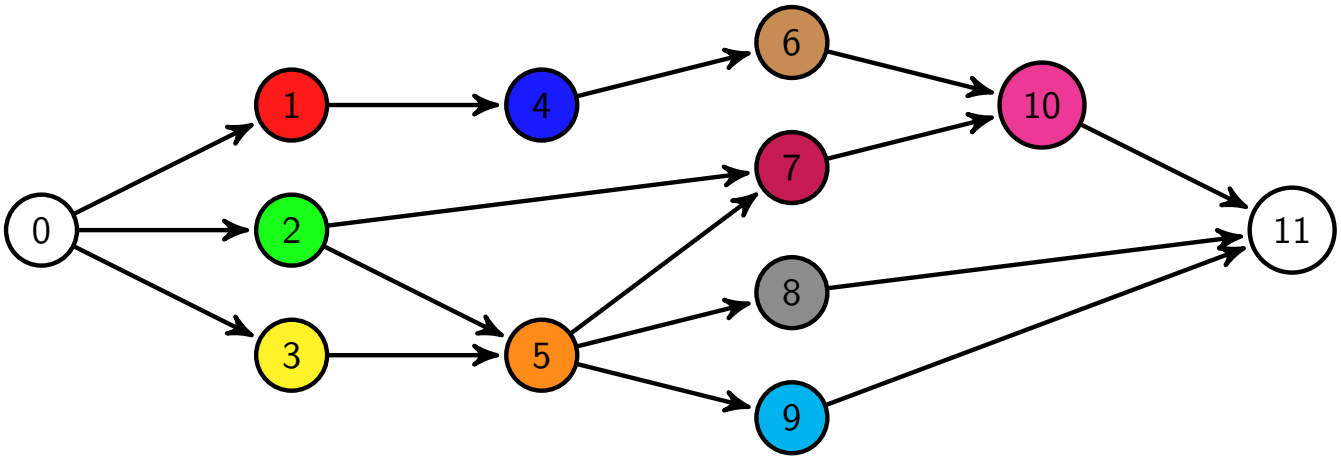
où $\mathcal{S}_T = \begin{cases} S_j \geq S_i + p_i & (i,j) \in E \quad \textit{Contraintes de précédence} \\ \sum_{j \in A(t)} b_{jk} \leq B_k & t \in \mathcal{T}, k \in R \quad \textit{Contraintes de ressources} \\ 0 \leq S_j \leq T - p_j & i \in A \end{cases}$

avec $A(t) = \{j \in A | t \in [S_j, S_j + p_j)\}, \forall t \in \mathcal{T}$

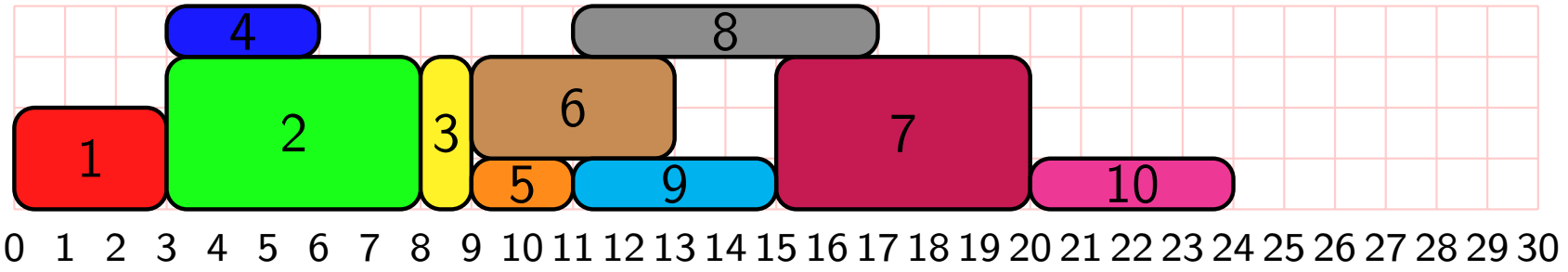
$\mathcal{S}_T^{\emptyset}$: ensemble des ordonnancements respectant les contraintes de précédence et l'horizon de temps T.

RCPSP: exemple de solution

$|R| = 1, B = 4, \mathcal{T} = [0, 30]$



i	p_i	b_i
1	3	2
2	5	3
3	1	3
4	3	1
5	2	1
6	4	2
7	5	3
8	6	1
9	4	1
10	4	1

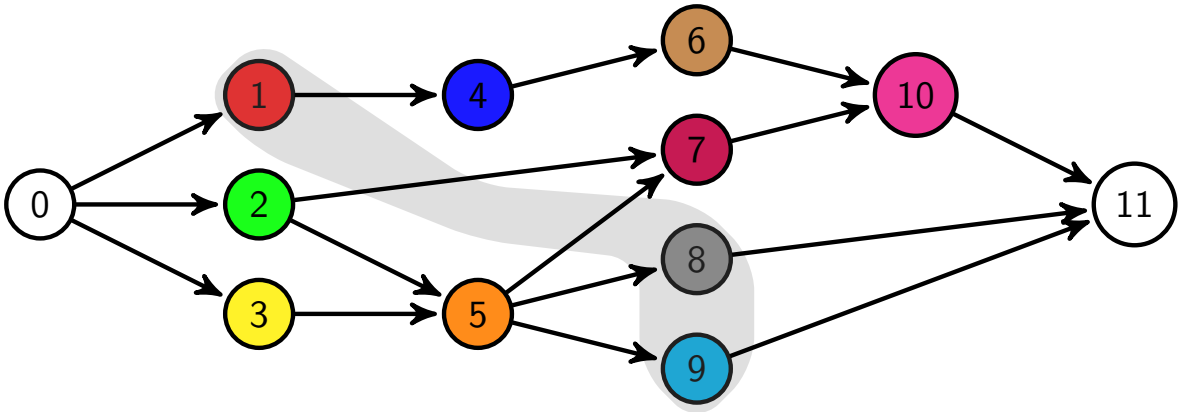


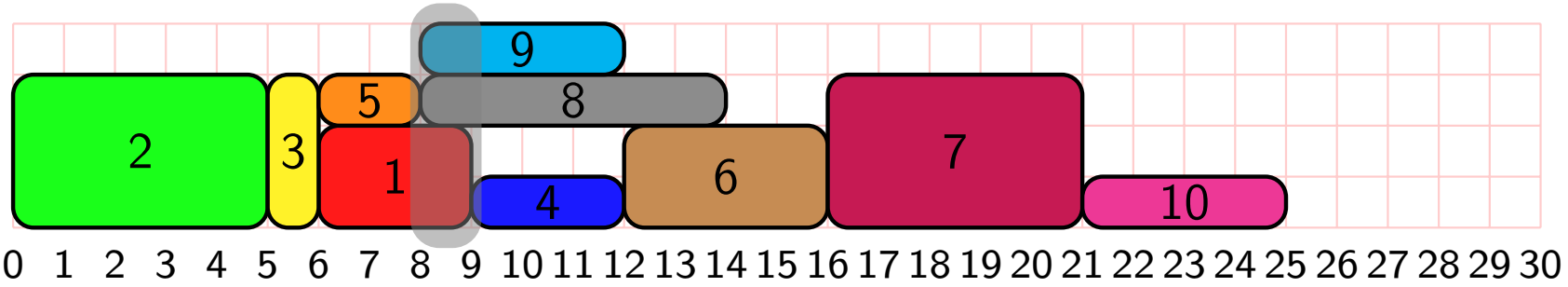
RCPSP: complexité, variantes and méthodes

- NP difficile au sens fort
- Généralisation des problèmes à une machine, machines parallèles, job-shop, open-shop, flow-shop
- Multitude de variantes
  - Autres objectifs: $\min \sum_{i \in A} w_i(S_i + p_i)$
  - Contraintes de précédence généralisées $S_j \geq S_i + l_{ij}$
  - Temps de préparation modes multiples, ressources consommables, tâches à intensités variables ...
  - Incertitude $p_i \in [p_i^{\min}, p_i^{\max}]$, $p_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$
- Méthodes exactes et approchées
  - Heuristiques et Metaheuristiques [Kolisch & Hartmann 2006, A. & Rivreau 2008]
  - Méthodes spécifiques de séparation et évaluation
  - Programmation linéaire en nombres entiers (MILP)
  - Programmation par contraintes (CP)
  - hybridations SAT/CP

Antichaînes

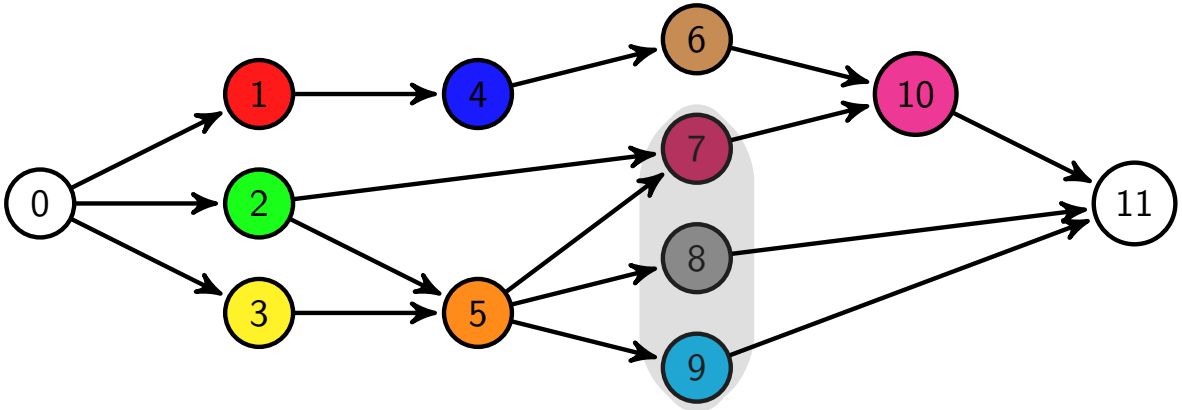
- Une antichaîne C de $G(A, E)$ est un ensemble de tâches non reliées deux à deux par un chemin.
- Pour toute antichaîne C, il existe une valeur T et un ordonnancement $S \in \mathcal{S}_T^{\emptyset}$ tels que $(\cap_{i \in C}[S_i, S_i + p_i) \neq \emptyset)$

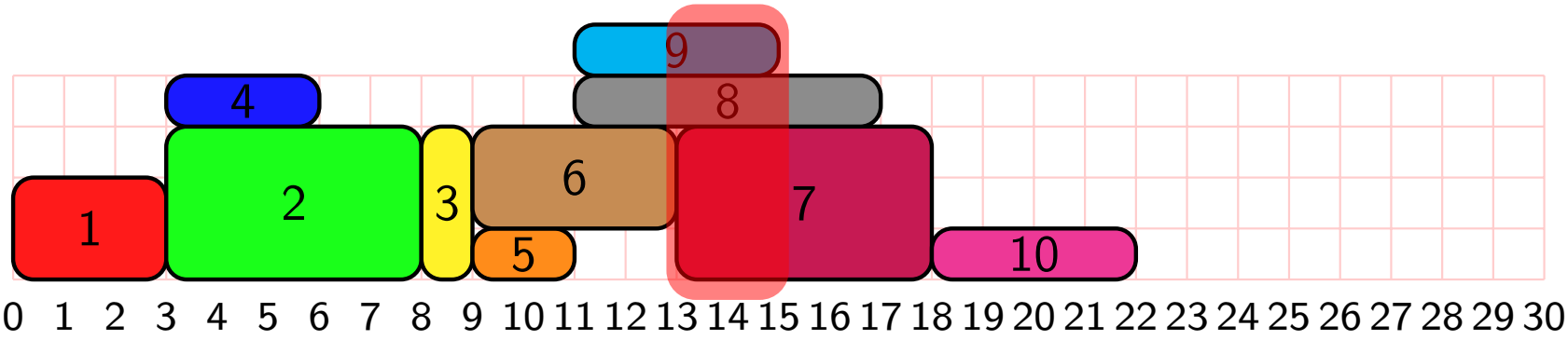




Antichaînes

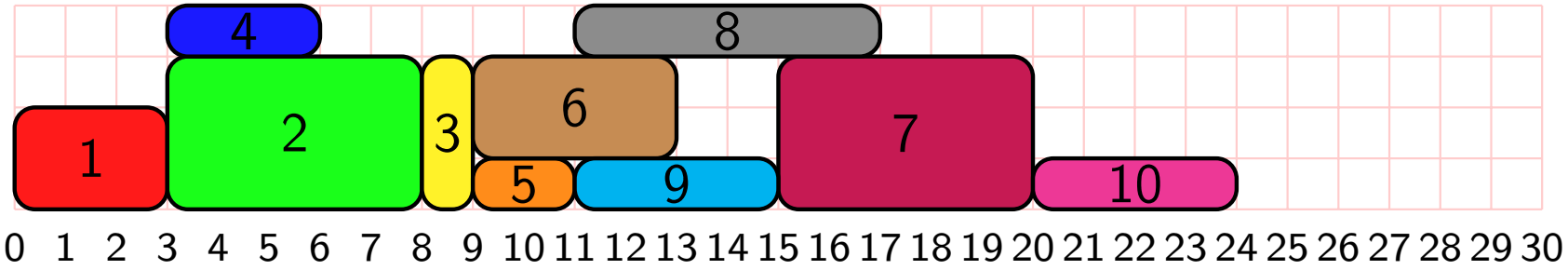
- Une antichaîne C de $G(A, E)$ est un ensemble de tâches non reliées deux à deux par un chemin.
- Pour toute antichaîne C, il existe une valeur T et un ordonnancement $S \in \mathcal{S}_T^{\emptyset}$ tels que $(\cap_{i \in C}[S_i, S_i + p_i) \neq \emptyset)$





Ensembles critiques minimaux

- Un ensemble critique F est une antichaîne de $G(V, E)$ telle que $\exists k \in R, \sum_{i \in F} b_{ik} > Bk$
- $\mathcal{MF}$: ensemble des ensembles critiques minimaux au sens de l'inclusion.



$$\mathcal{F} = \{\{1,2\},\{1,3\},\{1,7\},\{2,3\},\{2,6\},\{3,6\},\{6,7\},\{7,8,9\}\}$$

- Un ordonnancement $S \in \mathcal{S}_T^{\emptyset}$ est réalisable si et seulement si pour tout ensemble critique minimal F, $\exists i, j \in F$, $S_j \geq S_i + p_i$

Un ensemble critique est une sorte d'hyperarc disjonctif

Ordre strict et résolution des ensembles critiques minimaux

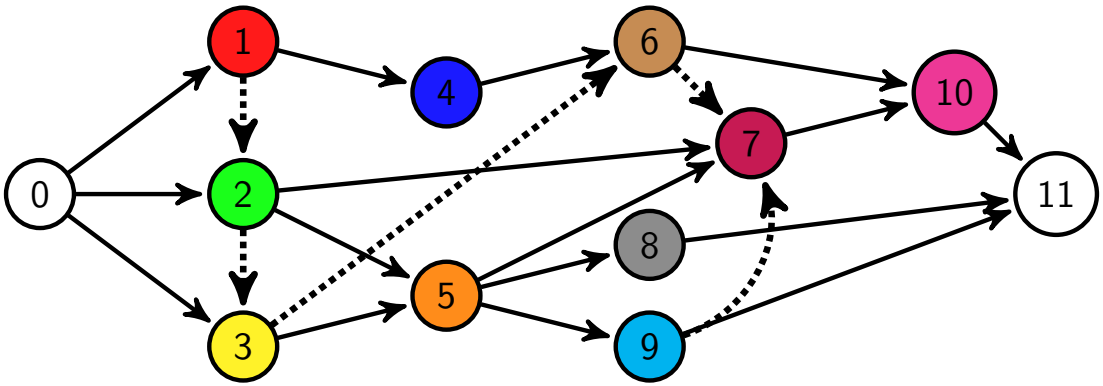
- X : un ensemble d'arcs «additionnels» : tel que $X \cap E = \emptyset$
- $G^X(A, E^+ \cup X, V)$: graphe potentiels-tâches incluant les arcs X.
- Soit $\prec^X$ la relation d'ordre strict sur A telle que:
  $i \prec^X j \Leftrightarrow$ «il existe dans G^X un chemin de longueur $\geq p_i$ de i vers j».
- $\mathcal{S}_T^X$ ensemble des ordonnancement respectant T et les contraintes de précédence $E \cup X$.
- On a $\mathcal{S}_T^X \subseteq \mathcal{S}_T$ si
  - $G(A, E^+ \cup X, V)$ n'a pas de circuit de longueur positive (1)
  - Pour tout $F \in \mathcal{MF}$, $\exists i, j \in F$ tel que $i \prec^X j$ (2).

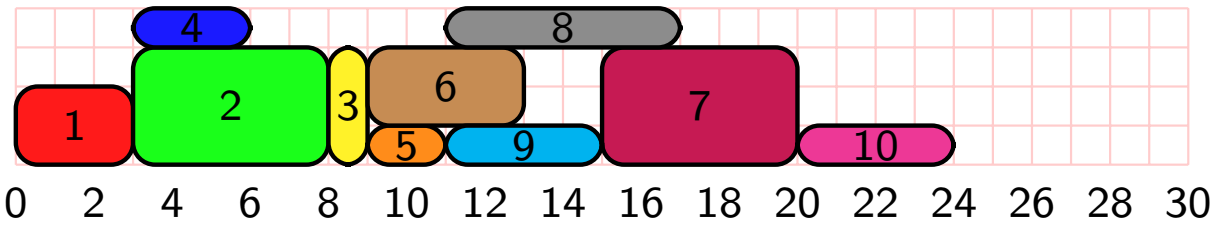
Représentation combinatoire de l'ensemble des solutions réalisables du RCPSP

Théorème : Représentation combinatoire de $\mathcal{S}_T$ [Bartusch et al. 1988]

Soit $\mathcal{X}$ l'ensemble des ensembles minimaux d'arcs qui vérifient (1) et (2).

$$\mathcal{S}_T = \cup_{X \in \mathcal{X}} \mathcal{S}_T^X \text{ (\textit{Union de polytopes convexes})}$$





$X = \{(1,2),(2,3),(3,6),(6,7),(9,7)\}$.

Tous les ensembles critiques minimaux sont résolus.

$\mathcal{F} = \{\{1,2\},\{1,3\},\{1,7\},\{2,3\},\{2,6\},\{3,6\},\{6,7\},\{7,8,9\}\}$

Ordonnancement au plus tôt ES^X de $\mathcal{S}_T^X$.

Algorithme

Dans un polytope $\mathcal{S}_T^X$, l'ordonnancement au plus tôt ES^X est dominant pour toute fonction objectif non décroissante.

$$(RCPSP) \min_{x \in \mathcal{X}} ES_{n+1}^X$$

On en déduit un algorithme de séparation et évaluation

Algorithme **BB-MFS**

1: $\mathcal{Q} \leftarrow \{X^0 = \emptyset\}$; $UB \leftarrow T$
2: **while** $\mathcal{Q} \neq \emptyset$ **do**
3: Prélever X de $\mathcal{Q}$
4: **Evaluer(**X,ES^X,LS^X,$UB-1$**)**
5: **if** $\mathcal{S}_{UB-1}^X \neq \emptyset$ **then**
6: **if** ES^X est réalisable et $ES_{n+1}^X < UB$ **then**
7: $UB \leftarrow ES_{n+1}^X$
8: **else**
9: sélectionner $F \in \mathcal{MF}$ violé par ES^X
10: **for** $i, j \in F, i < j$ **do**
11: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X \cup \{(i,j)\}\}, \{X \cup \{(j,i)\}\}$

Evaluer(X,ES^X,LS^X,$UB-1$**)**

- Réduction des fenêtres *propagation de contraintes*
- Calcul de borne inférieure spécifiques ou par *programmation linéaire*
- Application de règles de dominance

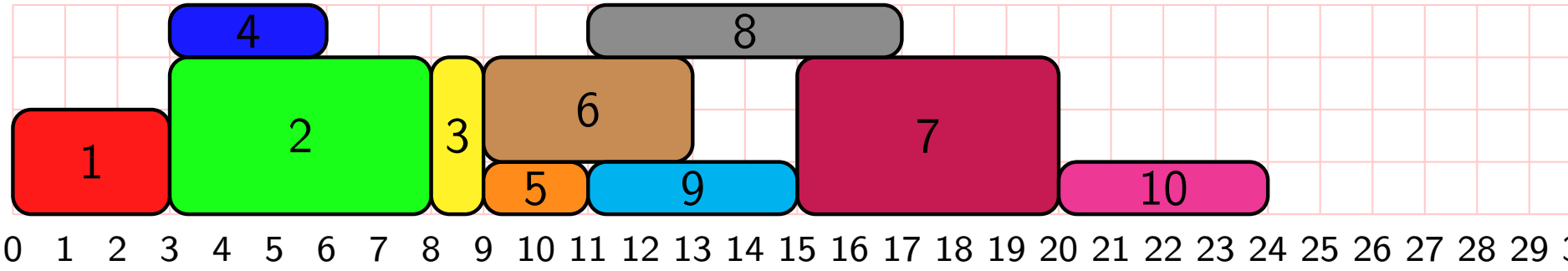
Formulations à temps continu basées sur les ensembles critiques

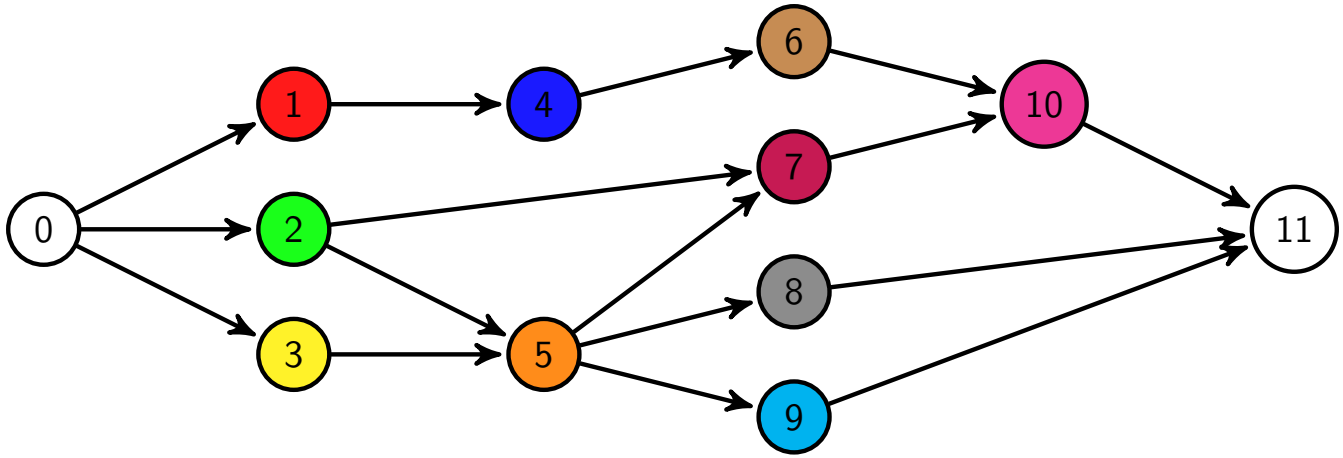
$$
\begin{aligned}
\text{Min. } & S_{n+1} \\
\text{s.t. } & z_{ij} + z_{ji} \leq 1 \quad i,j \in A,\ i < j \\
& z_{ij} + z_{jh} - z_{ih} \leq 1 \quad i,j,h \in A,\ i \neq j \neq h) \\
& z_{ij} = 1 \quad (i,j) \in E \\
& S_j - S_i + M_{ij}(1 - z_{ij}) \geq p_i \quad i,j \in A,\ i \neq j \\
& \sum_{i,j \in F, i \neq j} z_{ij} \geq 1 \quad F \in \mathcal{F} \\
& z_{ij} \in \{0,1\} \quad i,j \in A,\ i \neq j \\
& 0 \leq S_i \leq T - p_i \quad i \in A,\ i \neq j
\end{aligned}
$$

[Alvarez-Valdés and Tamarit 1993]
Extension de la formulation disjonctive du job-shop [Balas 1985]
Nombre exponentiel de contraintes

Le concept de flot d'unités de ressources

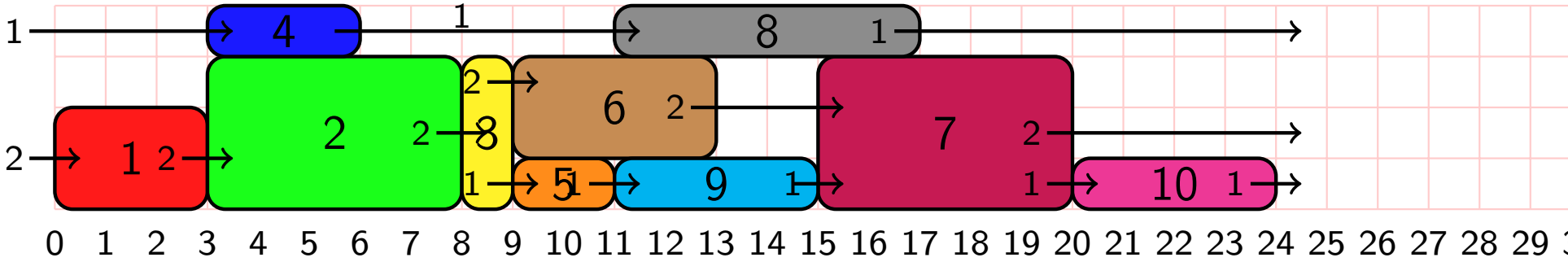
$\phi_{ij}^k \geq 0$: nombre d'unités de la ressource k transférées de i à j

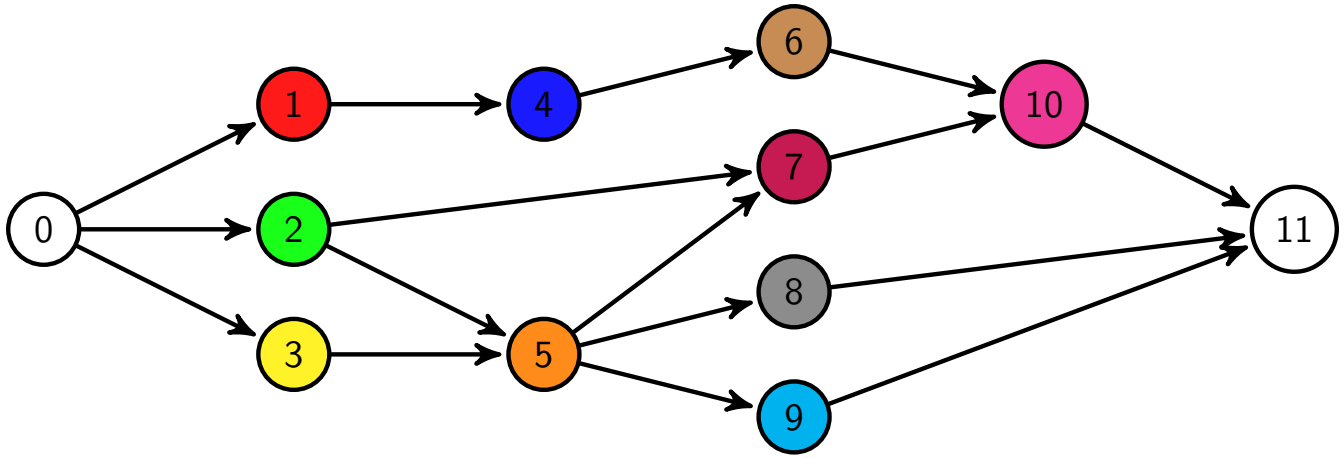




Le concept de flot d'unités de ressources

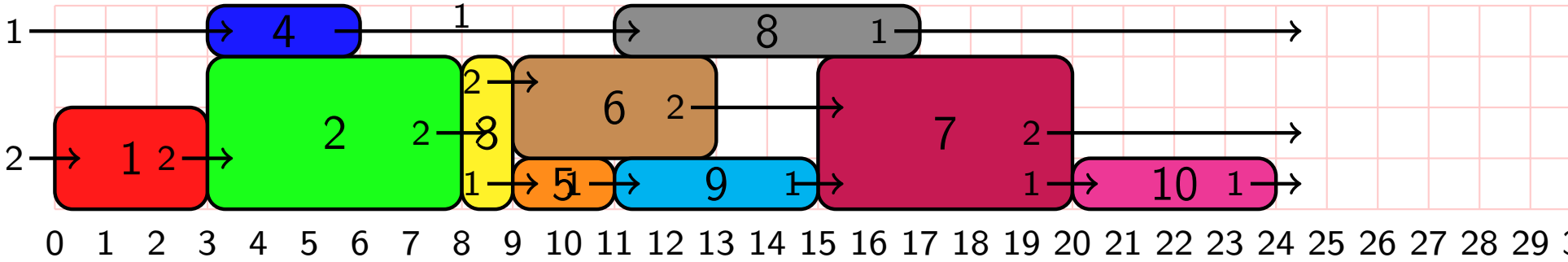
$\phi_{ij}^k \geq 0$: nombre d'unités de la ressource k transférées de i à j



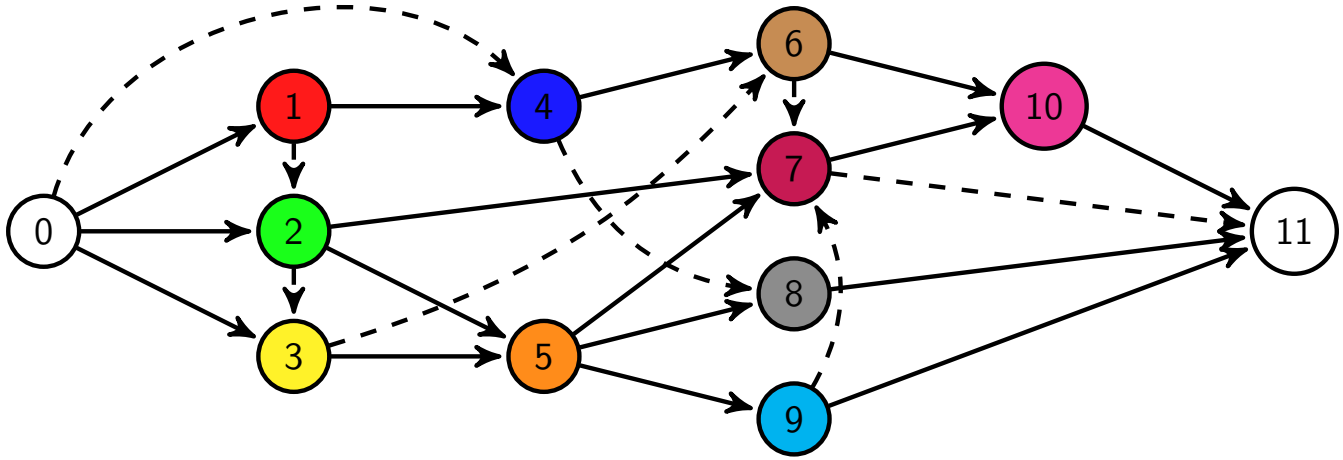


Le concept de flot d'unités de ressources

$\phi_{ij}^k \geq 0$: nombre d'unités de la ressource k transférées de i à j



Contrainte additionelle $\phi_{ij}^k > 0 \Rightarrow z_{ij} = 1$



Formulation basée sur les flots d'unités de ressource

- Remplacement des contraintes sur les ensembles critiques par :

$$\phi_{ij}^k - \min(\tilde{r}_{ik}, \tilde{r}_{jk}) z_{ij} \leq 0 \quad (i, j \in V,\ i \neq j,\ \forall k \in \mathcal{R})$$

$$\sum_{j \in V \setminus \{i\}} \phi_{ij}^k = \tilde{r}_{ik} \quad (i \in V \setminus \{n+1\})$$

$$\sum_{i \in V \setminus \{j\}} \phi_{ij}^k = \tilde{r}_{jk} \quad (j \in V \setminus \{0\})$$

$$0 \leq \phi_{ij}^k \leq \min(\tilde{r}_{ik}, \tilde{r}_{jk}) \quad (i, j \in V,\ i \neq n+1,\ j \neq 0,\ i \neq j;\ k \in \mathcal{R})$$

- $O(|A|^2 R)$ variables continues
- FB: une formulation compacte [A. *et al* 2003]

Formulation du problème d'insertion basée sur les flots (Resource-Constrained Activity insertion Problem)

Flot de référence

Un flot de référence est un flot valide pour $\mathcal{J} \setminus I$ tel que

- Aucun flot ne traverse I
- $S_j < S_i + p_i$ dans l'ordonnancement de référence $\Rightarrow f_{ijk} = 0$

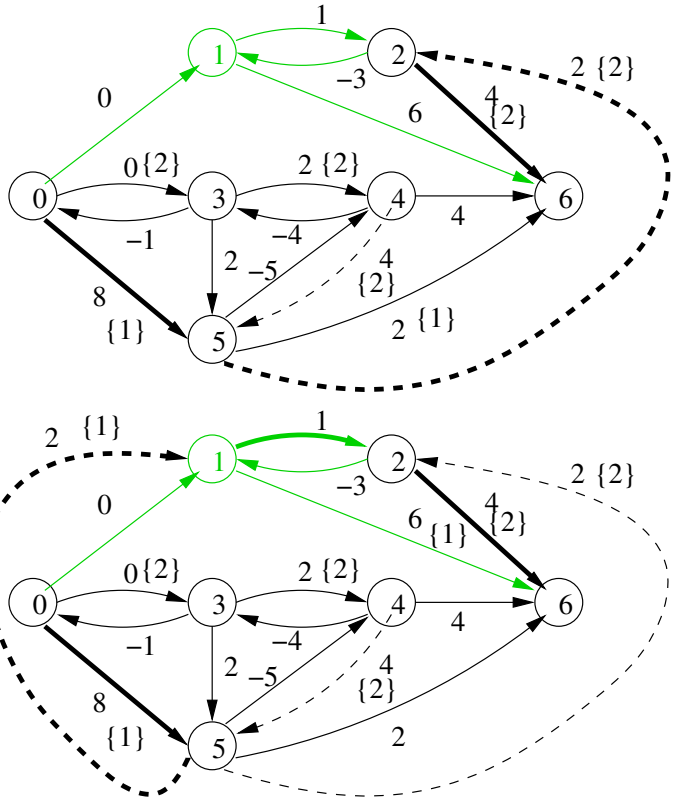
Graphe induit par le flot

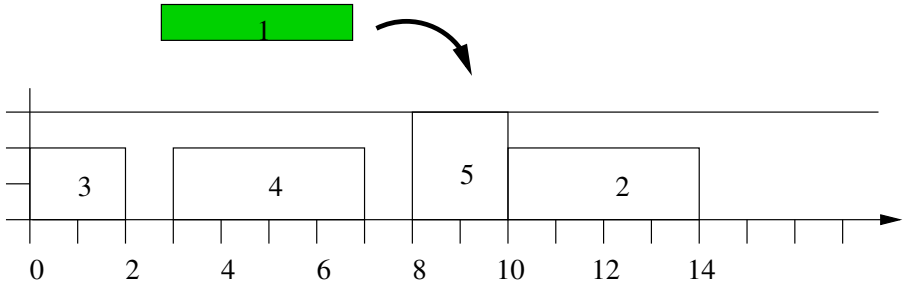
Graphe de précédence augmenté des arcs (i, j) valués par p_i pour chaque flot $f_{ij} > 0$.

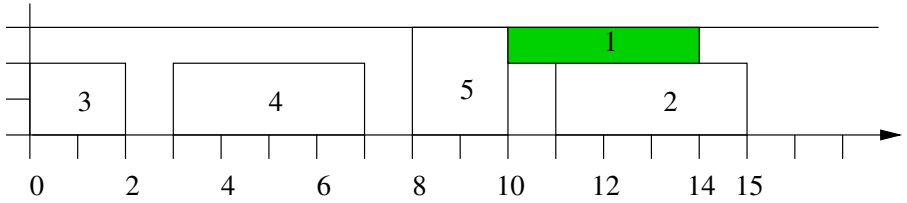
Formulation du RCAIP [A. et Briand 2009]

A partir du flot de référence f, trouver un flot f' valide pour A dont le graphe induit à une longueur de plus long chemin entre 0 et $n+1$ minimale **sans augmenter la valeur du flot entre deux activités de** $A \setminus I$.

Insertion d'une activité dans un RCPSP avec time lags







Remarque

Une position d'insertion peut être représentée par une coupe partielle (α, β) du graphe induit par l'ordonnancement de référence telle que $\sum_{i \in \alpha, j \in \beta} f_{ij} \geq b_i$

Caractérisation des insertions réalisables pour la variante décisionnelle du RCAIP

Variante décisionnelle

Variante telle que $C_{\max} \leq v$, valeur de l'arc $(n+1, 0) = \max(l_{(n+1)0}, -v)$.

Matrice des distances

- δ_{ij}^{v} est la longueur du plus long chemin de i à j dans le graphe induit par le flot pour une durée totale d'au plus v.
- δ^{v} se calcule en $O(n^3)$ par l'algorithme de Floyd-Warshall.

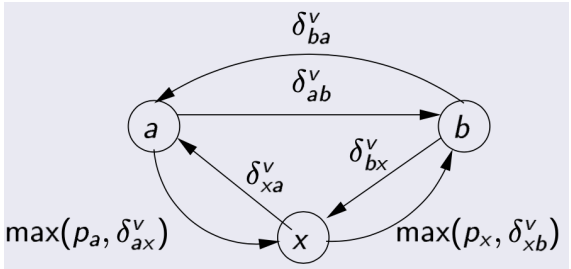
Condition nécessaire d'existence

Il n'y a pas de solution de durée d'au plus v si le problème de calcul de δ^{v} n'a pas de solution.

Caractérisation des insertions réalisables pour la variante décisionnelle du RCAIP

Insérer x dans (α, β) peut créer les circuits suivants ($a \in \alpha$, $b \in \beta$).

Circuits élémentaires induits par le flot



Theorem

L'insertion de x *dans* (α, β) *est possible si et seulement si* $\max(L_1, L_2, L_3) \leq 0$ *avec* $L_1 = \max_{a \in \alpha}(p_a + \delta^v_{xa})$, $L_2 = \max_{b \in \beta}(p_x + \delta^v_{bx})$, $L_3 = \max_{(a,b) \in \alpha \times \beta}(p_a + p_x + \delta^v_{ba})$.

Complexité du problème

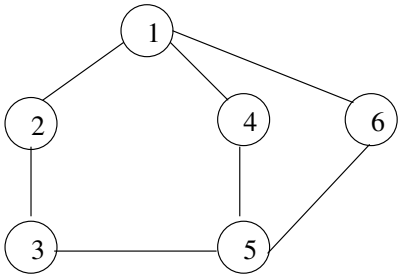
Theorem

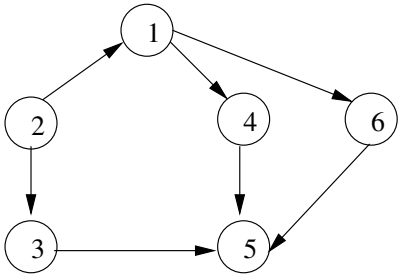
Le RCAIP est NP difficile

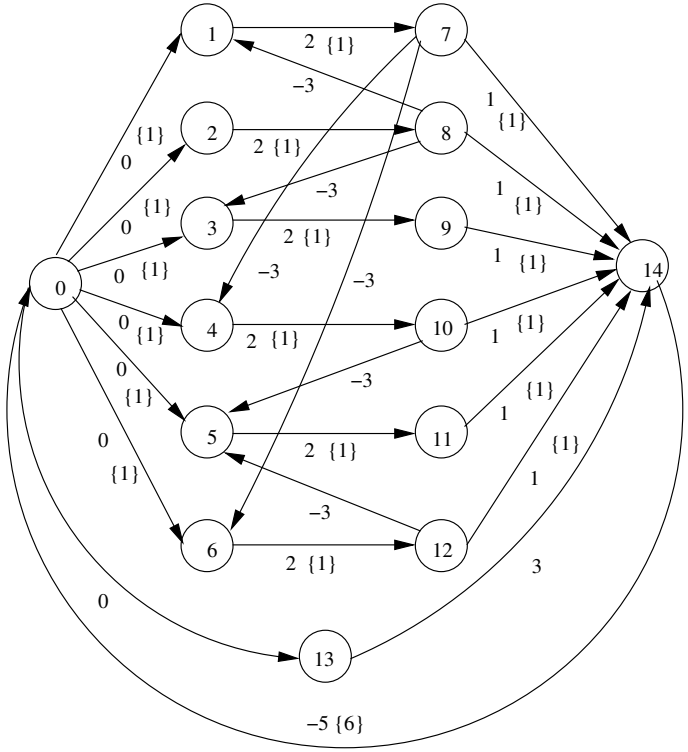
Preuve

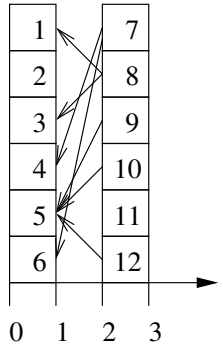
- Réduction du problème de stable.

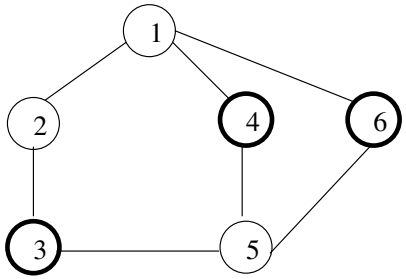
Preuve (illustration)

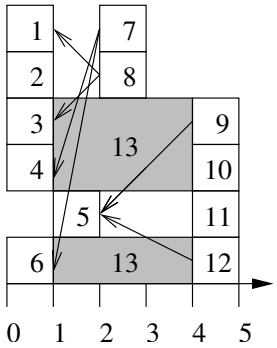








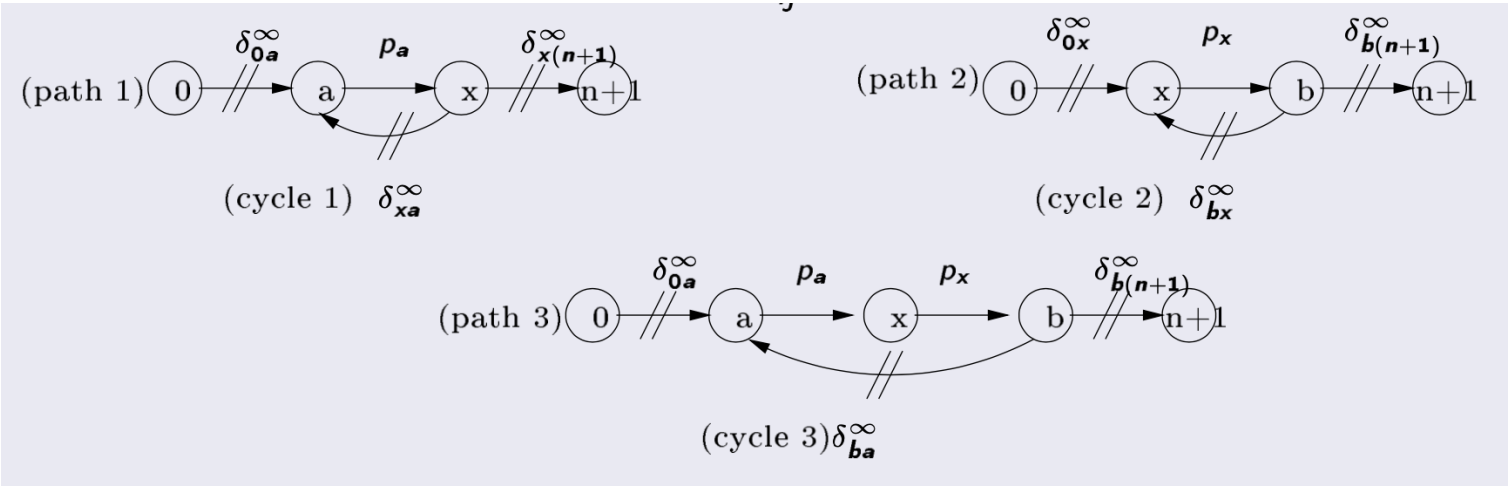




RCPSP avec écarts l_{ij} positifs (variante d'optimisation)

Circuits créés par l'insertion et plus longs chemins $0, n+1$

On considère la matrice des distances δ^{∞}_{ij}, $a \in \alpha$, $b \in \beta$



L'expression des plus longs chemins $0, n+1$ permet de définir des relations de dominance entre positions d'insertion.

Un algorithme d'insertion polynomial

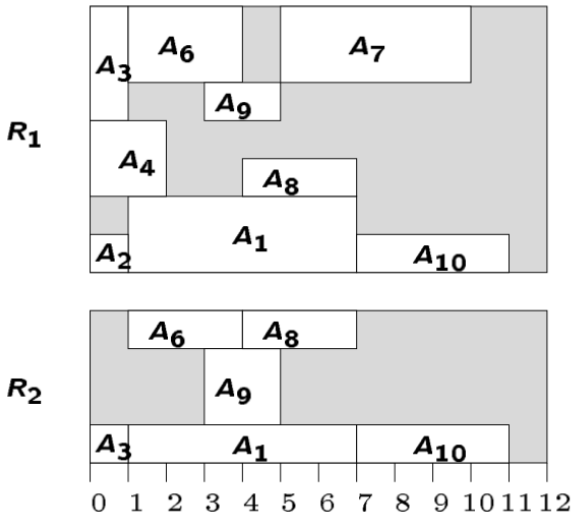
Définitions

- $\gamma_0 = \{i \in A | \delta_{ix}^{\infty} = 0 \text{ et } \delta_{xi}^{\infty} = 0\}$ est l'ensemble des activités synchronisées avec x.
- $\alpha_0 = \{i \in V \setminus \gamma_0 | \delta_{ix}^{\infty} \geq 0\}$
- $\beta_0 = V \setminus (\gamma_0 \cup \alpha_0)$
- $\mu(\alpha, \beta) = \{i \in \alpha | \delta_{0i}^{\infty} + p_i = \max_{A_a \in \alpha}(\delta_{0a}^{\infty} + p_a)\}$
- $\nu(\alpha, \beta) = \{i \in \beta | \delta_{i(n+1)}^{\infty} = \max_{b \in \beta}(\delta_{b(n+1)}^{\infty})\}$
- $\nu'(\alpha, \beta) = \{i \in \nu | \delta_{xi}^{\infty} = -\infty\}$

Algorithme $O(n^2 m)$

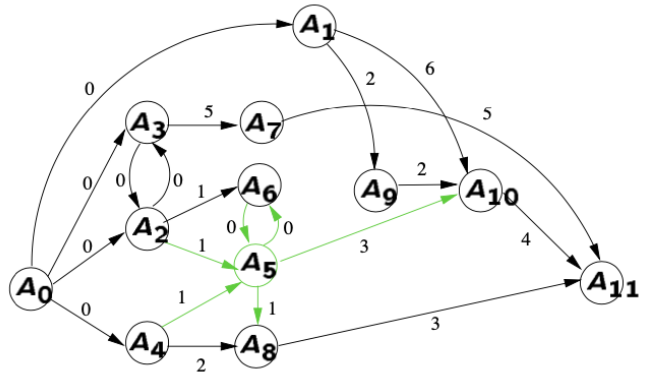
- Coupe initiale $(\alpha, \beta) \leftarrow (\alpha_0, \beta_0)$
- Tant qu'il reste assez de capacité dans (α, β) faire évoluer récursivement les coupes partielles obtenues en enlevant μ d'α puis passer à la coupe suivante $(\alpha, \beta) \leftarrow (\alpha \cup \nu', \beta \setminus \nu)$.

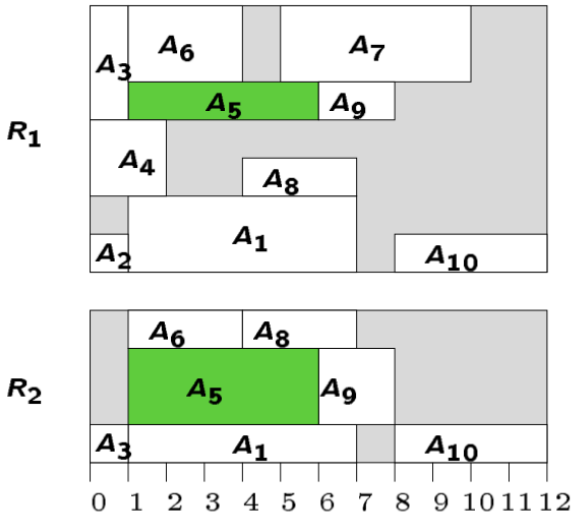
Exemple de RCPSP avec écarts minimaux seulement

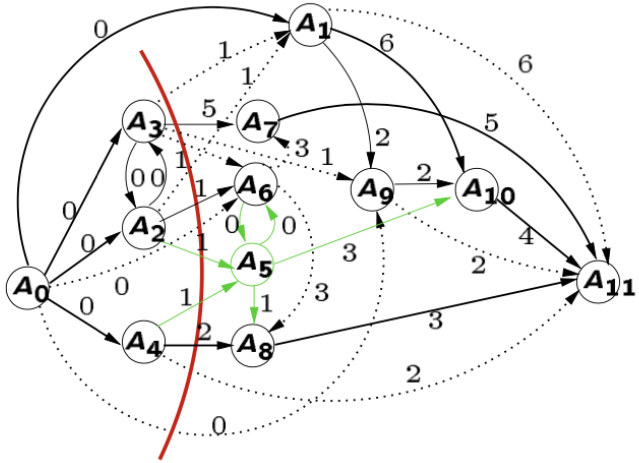


A_5

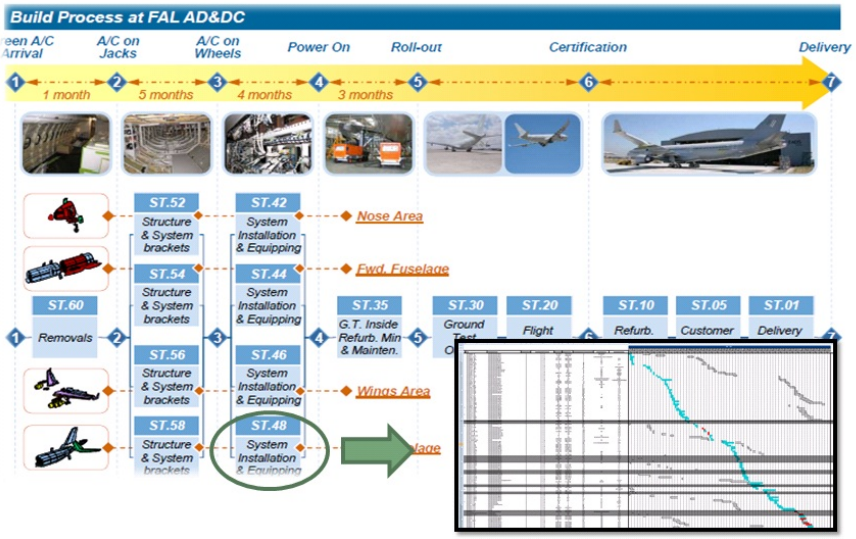
A_5







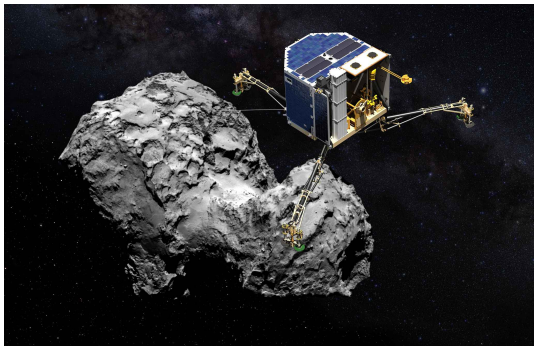
Exemple d'application
Chaine d'assemblage de l'Airbus A330 MRT

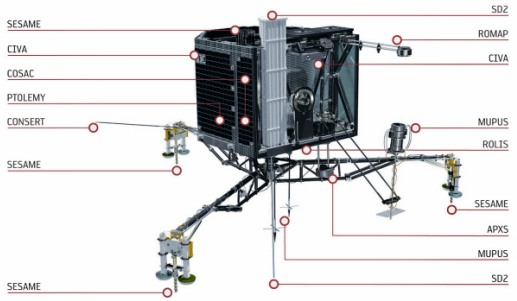


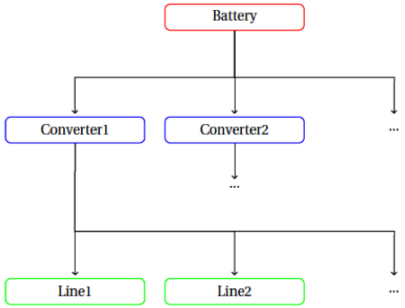
- RCPSP à date de fin imposée (*takt time*), objectif minimiser le nombre maximal de ressources utilisées
- Environ 600 tâches
- Ressources : opérateurs
- Problème multi-modes

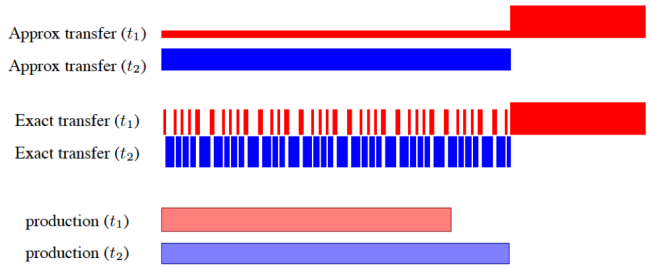
[Borreguero et al., 2015,2021]

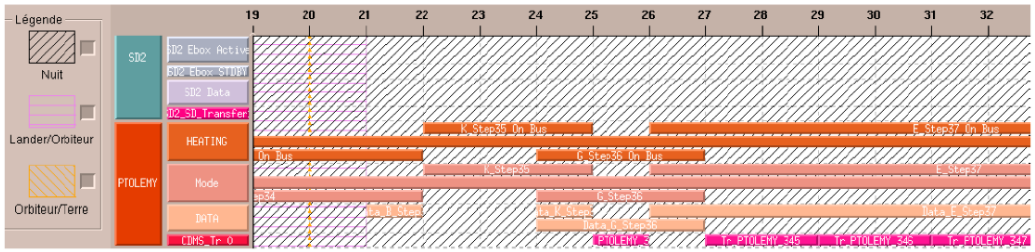
Exemple d'application : ordonnancement des expériences de Philae sur la comète «Tchouri»











- RCPSP avec contraintes de transferts de données
- Hiérarchie de ressources cumulatives à trois niveaux
- 19 expériences, 752 tâches, 168 événements, 926 précédences,

[Simonin et al., 2012 2015]

subsection in toc subsection in toc subsection in toc

Familles de formulations

Compromis concernant la taille des formulations et la qualité de la relaxation

- Formulations pseudo-polynomiales ou étendues $\rightarrow$ relaxations plus fortes, problèmes de temps et de mémoire, génération de colonnes
- Formulations compactes $\rightarrow$ relaxations plus faibles, inégalités valides

Classification de [Queyranne and Schulz 1994] selon le type de variables:

1. Variables continues de dates de début naturelles S_i et variables binaires d'ordre strict z_{ij}
2. Variables continues de dates d'événements t_e et variables binaires a_{ie} d'affectation aux événements
3. Variables binaires indexées par le temps x_{it}

Formulations à temps continu

Min. S_{n+1}

s.t. $z_{ij} + z_{ji} \leq 1 \quad i,j \in A,\ i < j$

$z_{ij} + z_{jh} - z_{ih} \leq 1 \quad i,j,h \in A,\ i \neq j \neq h)$

$z_{ij} = 1 \quad (i,j) \in E$

$S_j - S_i + M_{ij}(1 - z_{ij}) \geq p_i \quad i,j \in A,\ i \neq j$

$\phi_{ij}^k - \min(\tilde{r}_{ik}, \tilde{r}_{jk}) z_{ij} \leq 0 \quad (i,j \in V,\ i \neq j,\ \forall k \in \mathcal{R})$

$$\sum_{j \in V \setminus \{i\}} \phi_{ij}^k = \tilde{r}_{ik} \quad (i \in V \setminus \{n+1\})$$

$$\sum_{i \in V \setminus \{j\}} \phi_{ij}^k = \tilde{r}_{jk} \quad (j \in V \setminus \{0\})$$

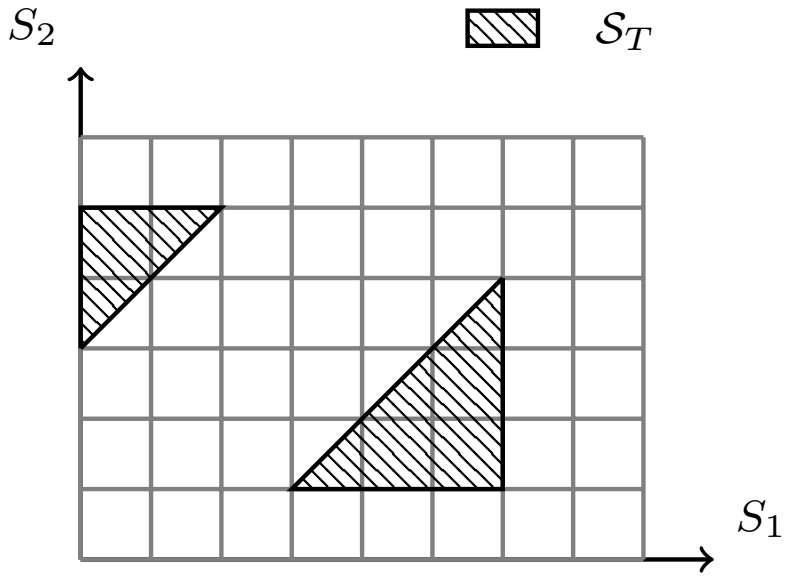
$0 \leq \phi_{ij}^k \leq \min(\tilde{r}_{ik}, \tilde{r}_{jk}) \quad (i,j) \in V,\ i \neq n+1,\ j \neq 0,\ i \neq j;\ k \in \mathcal{R}$

$z_{ij} \in \{0,1\} \quad i,j \in A,\ i \neq j$

$0 \leq S_i \leq T - p_i \quad i \in A,\ i \neq j$

Relaxation de mauvaise qualité

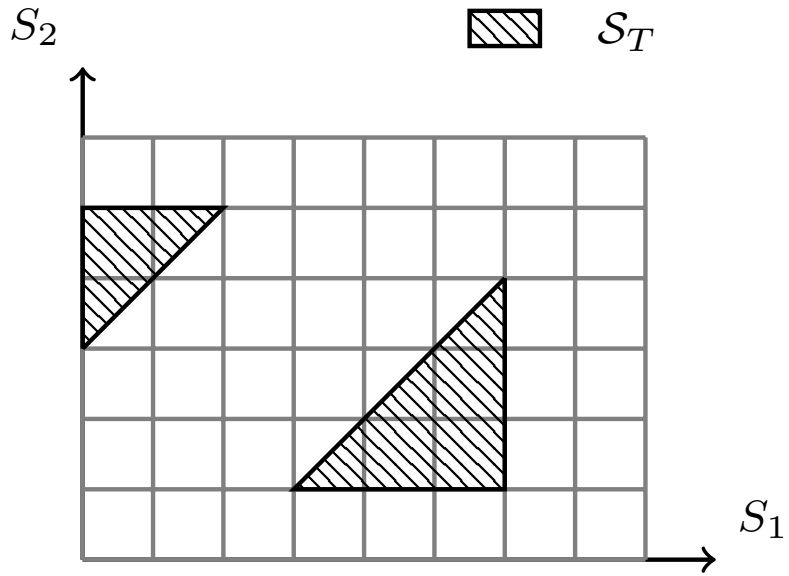
- Exemple 2 tâches, $ES = (0, 1), LS = (6, 5)$, un ensemble critique $\{1, 2\}$, minimisation de la somme des dates de fin



Relaxation de mauvaise qualité

- Exemple 2 tâches, $ES = (0, 1), LS = (6, 5)$, un ensemble critique $\{1, 2\}$, minimisation de la somme des dates de fin

$$
\begin{aligned}
(P)\ \min\ & S_1 + S_2 + 5 \\
& S_1 \geq 0 \\
& S_2 \geq 1 \\
& S_1 \leq 6 \\
& S_2 \leq 5 \\
& S_2 - S_1 + 8x \geq 3 \\
& S_1 - S_2 + 7(1 - x) \geq 2 \\
& x \in \{0, 1\}
\end{aligned}
$$

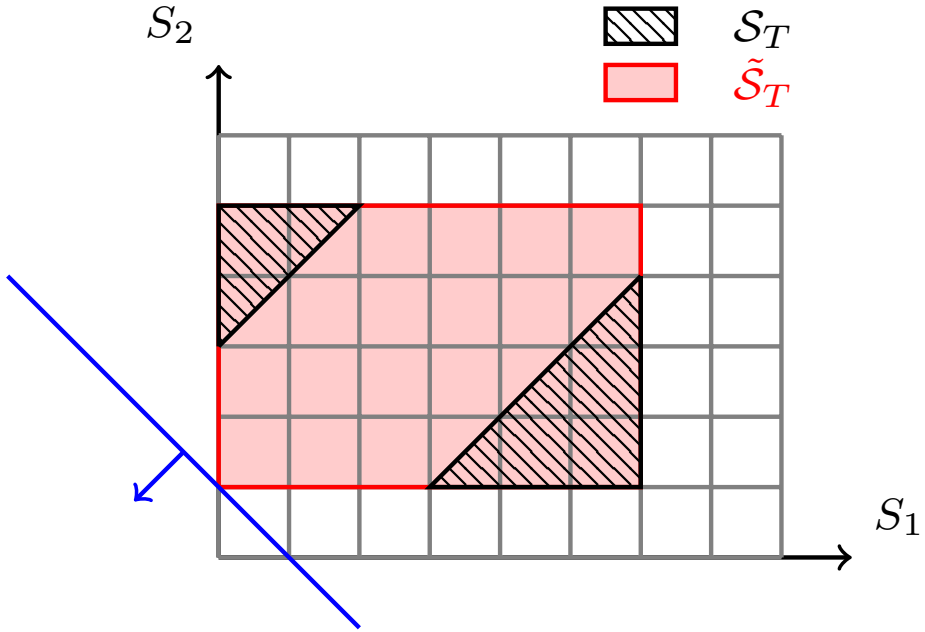


L'ensemble des solutions réalisable du PLNE est bien $\mathcal{S}_T$

Relaxation de mauvaise qualité

- Exemple 2 tâches, $ES = (0, 1), LS = (6, 5)$, un ensemble critique $\{1, 2\}$, minimisation de la somme des dates de fin

$$
\begin{aligned}
(P)\ \min\ & S_1 + S_2 + 5 \\
& S_1 \geq 0 \\
& S_2 \geq 1 \\
& S_1 \leq 6 \\
& S_2 \leq 5 \\
S_2 - S_1 + 8x & \geq 3 \\
S_1 - S_2 + 7(1 - x) & \geq 2 \\
x & \in \{0, 1\}
\end{aligned}
$$

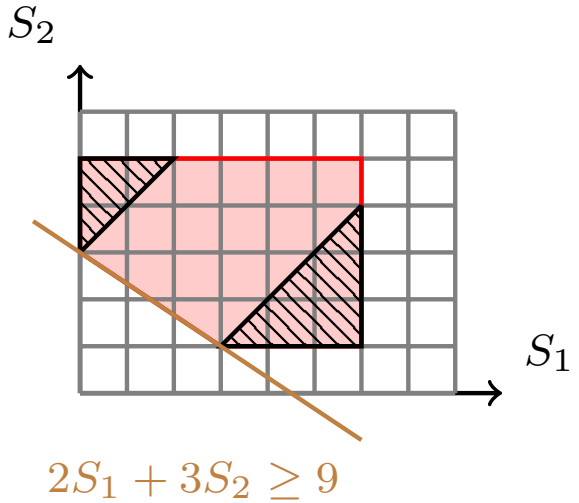


relaxation LB=6

pb : $x = 0.5$ toujours possible

Inégalités valides

- Se rapprocher de $conv(\mathcal{S}_T)$
- Exemple 1: Extension des inégalités valides pour le jobshop [Balas 85,Applegate & Cook 1991,Dyer & Wolsey 1990] (half-cuts, late job cuts...)



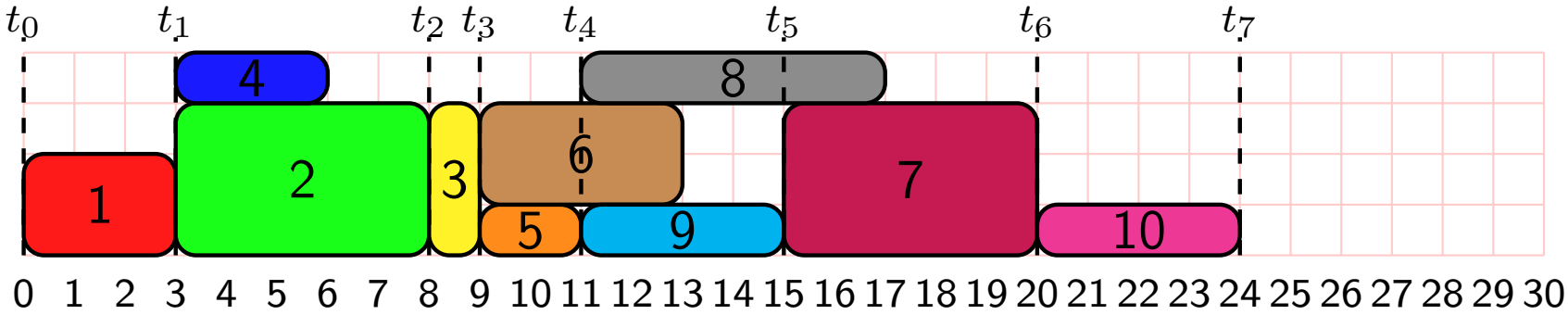
$$(p_i + ES_i - ES_j)S_i + (o_j + ES_j - ES_i) \geq p_i p_j + ES_i p_j + ES_j p_i$$

- Exemple 2: coupes basées sur la propagation de contraintes [Demassey *et al* 2005]
  - Distances conditionnelles $d_{ij}^{k \prec l}$, $d_{ij}^{l \prec k}$ and $d_{ij}^{k||l}$
  - Inégalités de distance liftées

$$S_j - S_i \geq d_{ij}^{h||l} + (d_{ij}^{h \prec l} - d_{ij}^{h||l})z_{hl} + (d_{ij}^{l \prec h} - d_{ij}^{h||l})z_{lh}$$

Formulation on/off basée sur les événements

- $\mathcal{E}$: ensemble d'événements ordonnés.
- $t_e \geq 0$: date d'événement: début de tâche
- Variable binaire on/off $a_{ie} = 1 \Leftrightarrow [S_i, S_i + p_i] \cap [t_e, t_e + 1] \neq \emptyset$
- Chaque tâche telle que $a_{ie} = 1$ est supposée couvrir $[t_e, t_e + 1]$
- $n|\mathcal{E}|$ variables binaires $\rightarrow$ formulation compacte



Extension des modèles sur le problèmes de machines [Lasserre and Queyranne 1992,Dauzère-Pérès and Lasserre 1995], et en ordonnancement de procédés [Pinto and Grossmann 1995, Zapata *et al* 2008].

(OOE) Min. $C_{\max}$

s. t. $C_{\max} \geq t_e + (a_{ie} - a_{i(e-1)})p_i \quad (e \in \mathcal{E};\ i \in A)$

$t_0 = 0$

$t_{e+1} \geq t_e \quad (e \neq n-1 \in \mathcal{E})$

$t_f \geq t_e + (a_{ie} - a_{i,e-1} - a_{if} + a_{i,f-1} - 1)p_i \quad ((e,f,i) \in \mathcal{E}^2 \times A,\ f > e \neq 0)$

$\sum_{e'=0}^{e-1} a_{ie'} \geq e(1 - a_{ie} + a_{i,e-1})) \quad (i \in A; e \neq 0 \in \mathcal{E})$

$\sum_{e'=e}^{n-1} a_{ie'} \geq e(1 + a_{ie} - a_{i,e-1}) \quad (i \in A; e \neq 0 \in \mathcal{E})$

$\sum_{e \in \mathcal{E}} a_{ie} \geq 1 \quad (i \in A)$

$a_{ie} + \sum_{e'=0}^{e} a_{je'} \leq 1 + (1 - a_{ie})e \quad (e \in \mathcal{E};\ (i,j) \in E)$

$\sum_{i=0}^{n-1} r_{ik} a_{ie} \leq R_k \quad (e \in \mathcal{E};\ k \in \mathcal{R})$

$t_e \geq 0 \quad (e \in \mathcal{E})$

$a_{ie} \in \{0,1\} \quad (i \in A;\ e \in \mathcal{E})$

[Koné et al. 2011, 2013]

Inégalités valides pour les modèles à événement

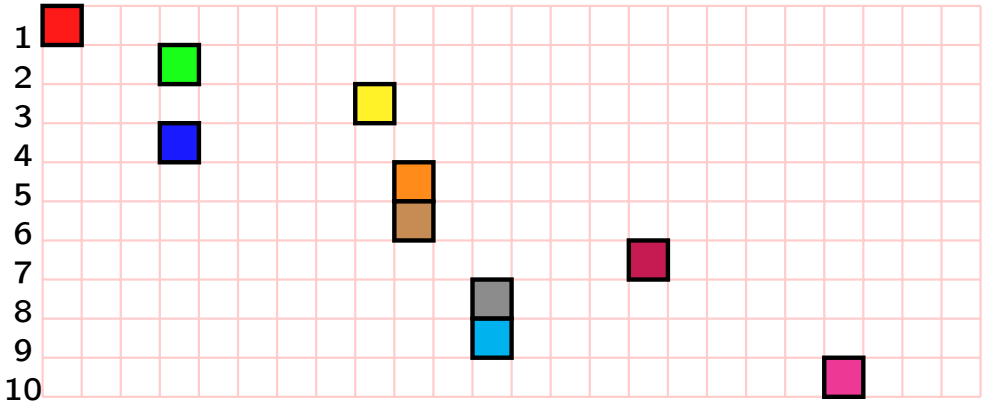
- Relaxation également de mauvaise qualité pour le RCPSP [Koné *et al.*. 2011]
- Améliorations par [Tesh 2020]
- Inégalités valides sur le modèle on-off [Nattaf, Kis, A., Lopez 2019]
  - Contraintes de non préemption
  - Description de l'enveloppe convexe du polytope des vecteurs de type 0001110000, 00110000, 000010, etc.

$$\sum_{e_k \in S} (-1)^k a_{ie_k} \leq 1 \quad \forall S = \{e_0, \ldots, e_{2l}\}, \text{ordered subset of events}$$

  - Nombre exponentiel d'inégalités mais algorithme polynomial de séparation

Variables indexées par le temps de type «impulsion»

- Pour des données entières, $\mathcal{S}_T$ peut être restreint à ses points entiers $\mathcal{S}_T^{\text{int}}$.
- Variable binaire impulsionnelle $x_{it} = 1 \Leftrightarrow S_i = t$, pour $t \in \mathcal{T} \cap \mathbb{N}$
- Nombre pseudo-polynomial de variables $|A||T|$



La formulation agrégée indexée par le temps (DT)

- $S_i = \sum_{t \in T} t\, x_{it}$
- $A(t) = \{i \in A | \exists \tau \in \{t - p_i + 1, \ldots, t\}, x_{i\tau} = 1\}$

$$(DT)\ \text{Min.} \sum_{t=0}^{T} t x_{n+1,t}$$

$$\text{s. c.} \quad \sum_{t=0}^{T} t x_{jt} - \sum_{t \in H} t x_{it} \geq p_i \quad (i,j) \in E$$

$$\sum_{i \in V} \sum_{\tau = t - p_i + 1}^{t} b_{ik} x_{i\tau} \leq B_k \quad t = 0, \ldots, T;\ k \in \mathcal{R}$$

$$\sum_{t=0}^{T} x_{it} = 1 \quad i \in A$$

$$x_{it} \in \{0,1\} \quad i \in A; t = 0, \ldots, T$$

[Pritsker et al. 1969]

Retour au petit exemple, une meilleure relaxation...

$$(P)\ \min S_1 + S_2 + 5$$

$$S_1 = x_{1,1} + 2x_{1,2} + 3x_{1,3} + 4x_{1,4} + 5x_{1,5} + 6x_{1,6}$$

$$S_2 = x_{2,1} + 2x_{2,2} + 3x_{2,3} + 4x_{2,4} + 5x_{2,5}$$

$$x_{1,0} + x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 1$$

$$x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} + x_{2,5} = 1$$

$$x_{1,0} + x_{1,1} + x_{2,1} \leq 1$$

$$x_{2,1} + x_{2,2} + x_{1,0} + x_{1,1} + x_{1,2} \leq 1$$

$$x_{2,2} + x_{2,3} + x_{1,1} + x_{1,2} + x_{1,3} \leq 1$$

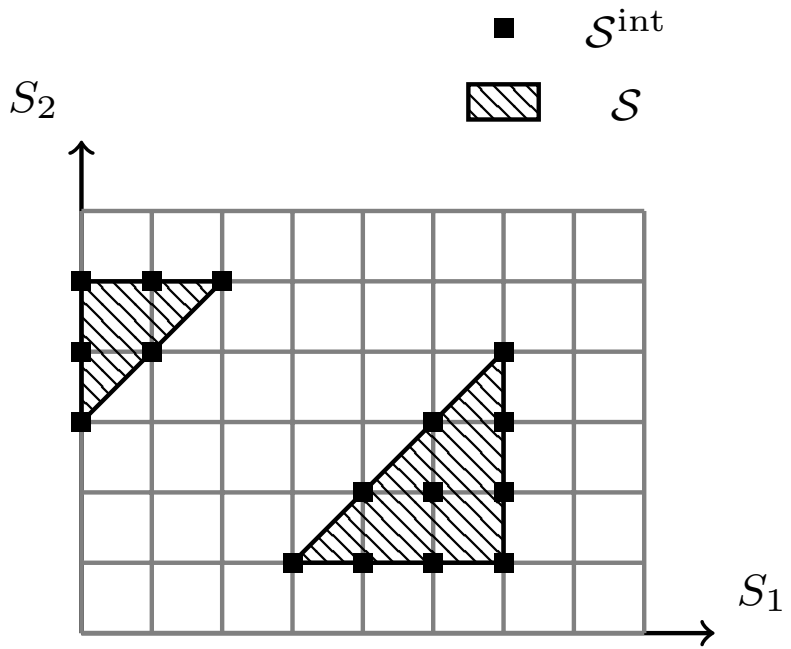
$$x_{2,3} + x_{2,4} + x_{1,2} + x_{1,3} + x_{1,4} \leq 1$$

$$x_{2,4} + x_{2,5} + x_{1,3} + x_{1,4} + x_{1,5} \leq 1$$

$$x_{2,5} + x_{1,4} + x_{1,5} + x_{1,6} \leq 1$$

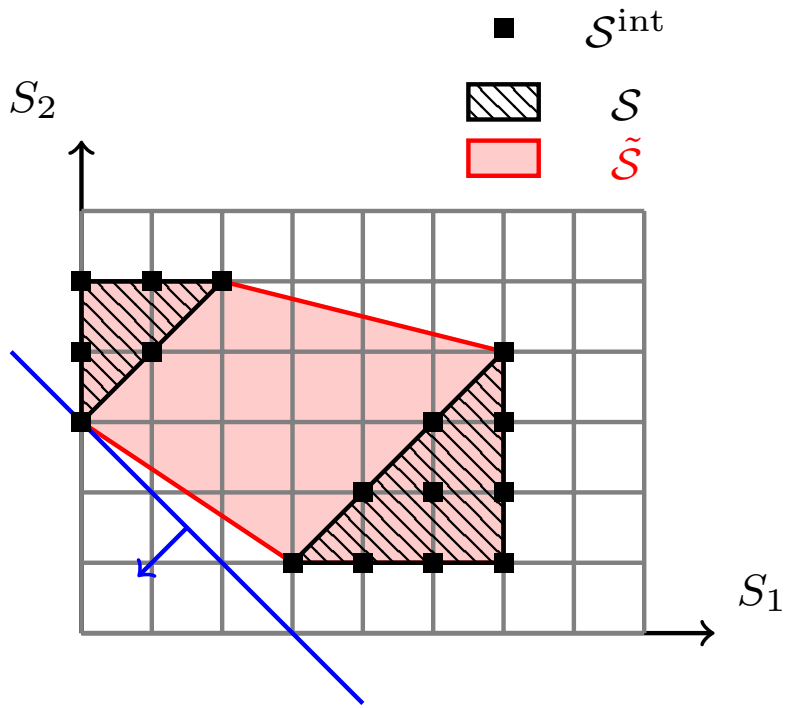
$$x_{1,t} \in \{0,1\} \quad t \in \{0,\ldots,6\}$$

$$x_{2,t} \in \{0,1\} \quad t \in \{1,\ldots,5\}$$



Retour au petit exemple, une meilleure relaxation...

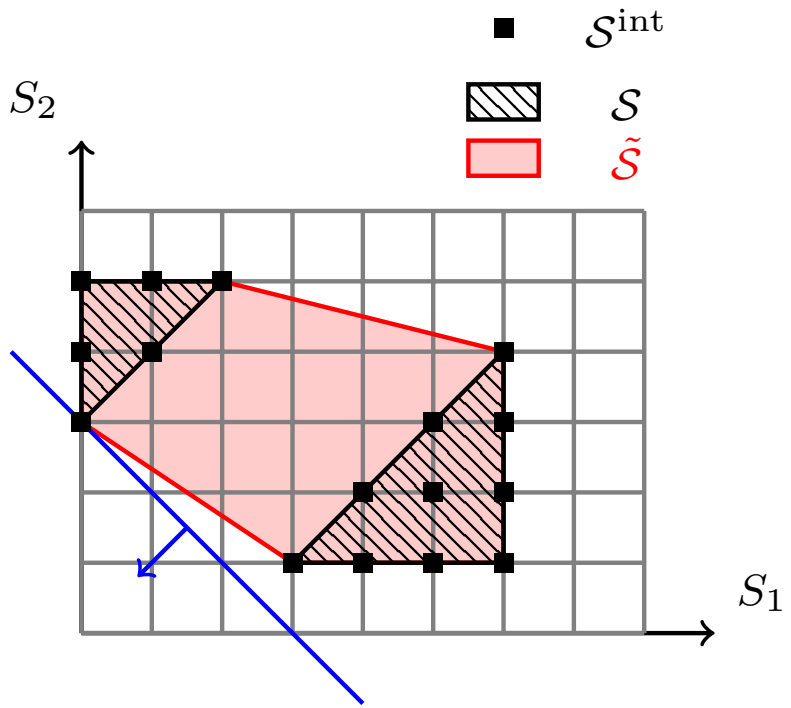
$$(P)\ \min S_1 + S_2 + 5$$
$$S_1 = x_{1,1} + 2x_{1,2} + 3x_{1,3} + 4x_{1,4} + 5x_{1,5} + 6x_{1,6}$$
$$S_2 = x_{2,1} + 2x_{2,2} + 3x_{2,3} + 4x_{2,4} + 5x_{2,5}$$
$$x_{1,0} + x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 1$$
$$x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} + x_{2,5} = 1$$
$$x_{1,0} + x_{1,1} + x_{2,1} \leq 1$$
$$x_{2,1} + x_{2,2} + x_{1,0} + x_{1,1} + x_{1,2} \leq 1$$
$$x_{2,2} + x_{2,3} + x_{1,1} + x_{1,2} + x_{1,3} \leq 1$$
$$x_{2,3} + x_{2,4} + x_{1,2} + x_{1,3} + x_{1,4} \leq 1$$
$$x_{2,4} + x_{2,5} + x_{1,3} + x_{1,4} + x_{1,5} \leq 1$$
$$x_{2,5} + x_{1,4} + x_{1,5} + x_{1,6} \leq 1$$
$$x_{1,t} \in \{0,1\} \quad t \in \{0,\ldots,6\}$$
$$x_{2,t} \in \{0,1\} \quad t \in \{1,\ldots,5\}$$



Dans cet exemple $\tilde{\mathcal{S}}_T = conv(\mathcal{S}_T)$, la relaxation est idéale.

Retour au petit exemple, une meilleure relaxation...

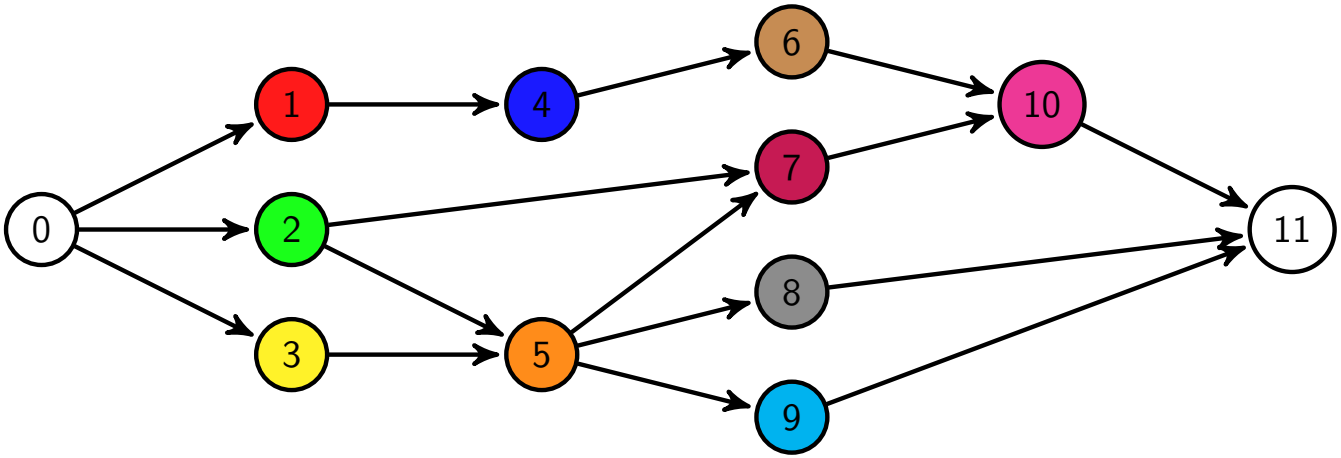
$$(P)\ \min S_1 + S_2 + 5$$
$$S_1 = x_{1,1} + 2x_{1,2} + 3x_{1,3} + 4x_{1,4} + 5x_{1,5} + 6x_{1,6}$$
$$S_2 = x_{2,1} + 2x_{2,2} + 3x_{2,3} + 4x_{2,4} + 5x_{2,5}$$
$$x_{1,0} + x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 1$$
$$x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} + x_{2,5} = 1$$
$$x_{1,0} + x_{1,1} + x_{2,1} \leq 1$$
$$x_{2,1} + x_{2,2} + x_{1,0} + x_{1,1} + x_{1,2} \leq 1$$
$$x_{2,2} + x_{2,3} + x_{1,1} + x_{1,2} + x_{1,3} \leq 1$$
$$x_{2,3} + x_{2,4} + x_{1,2} + x_{1,3} + x_{1,4} \leq 1$$
$$x_{2,4} + x_{2,5} + x_{1,3} + x_{1,4} + x_{1,5} \leq 1$$
$$x_{2,5} + x_{1,4} + x_{1,5} + x_{1,6} \leq 1$$
$$x_{1,t} \in \{0,1\} \quad t \in \{0,\ldots,6\}$$
$$x_{2,t} \in \{0,1\} \quad t \in \{1,\ldots,5\}$$



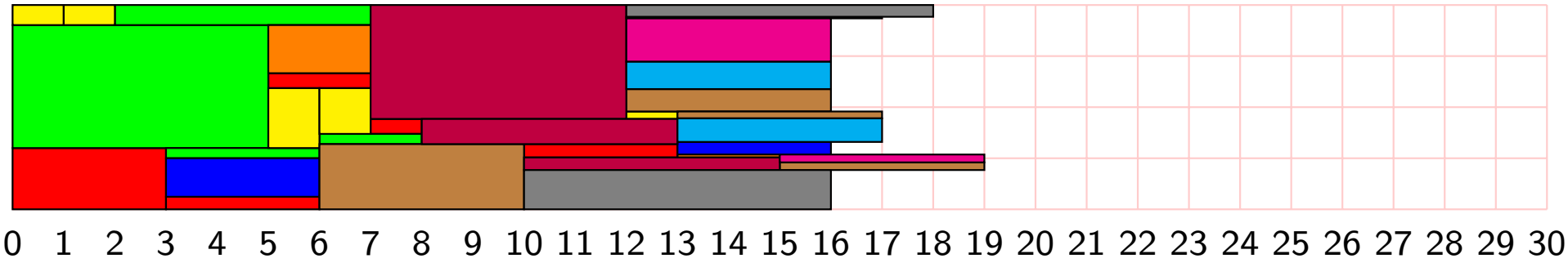
Dans cet exemple $\tilde{\mathcal{S}}_T = conv(\mathcal{S}_T)$, la relaxation est idéale.
... on a besoin de 11 variables binaires pour 2 tâches

... pas si bon en general

$|R| = 1, B = 4, \mathcal{T} = [0, 30]$



i	p_i	b_i
1	3	2
2	5	3
3	1	3
4	3	1
5	2	1
6	4	2
7	5	3
8	6	1
9	4	1
10	4	1



Borne inférieure = 16.46 (17) (pas meilleur que la borne triviale)

La formulation désagrégée (DDT)

Les contraintes de précédence peuvent être désagrégées :

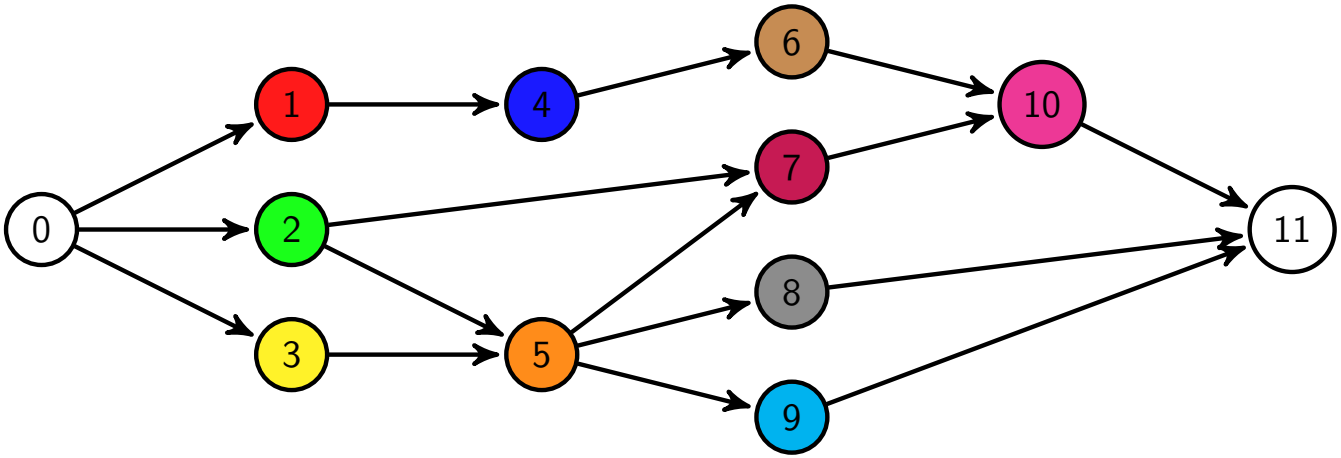
$$\sum_{\tau=0}^{t-p_i} x_{i\tau} - \sum_{\tau=0}^{t} x_{j\tau} \geq 0 \quad (i,j) \in E;\ t \in T$$

[Christofides *et al.* 1997]

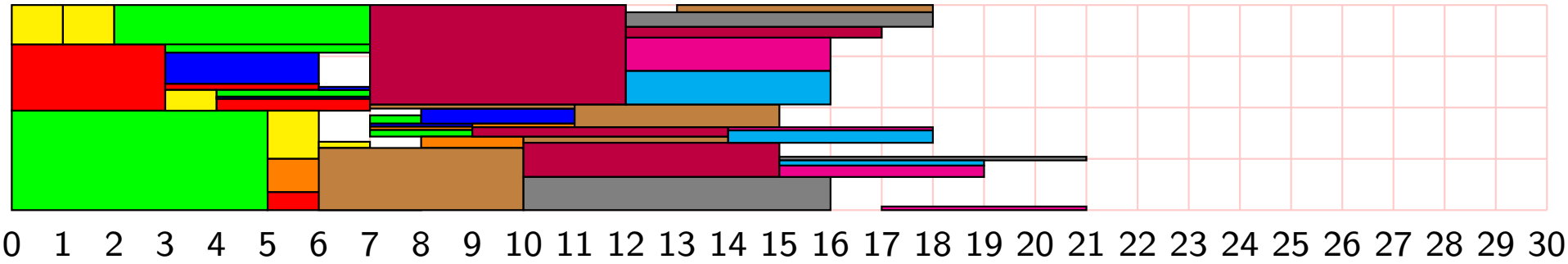
- Relation logique: $S_j \leq t \Rightarrow S_i \leq t - p_i$
- La matrice sans les contraintes de ressources est totalement unimodulaire (TU).
- La relaxation lagrangienne des contraintes de ressources préserve TU. Calcul efficace par un algo. de flot max [Möhring *et al.* 2003]

DDT: qualité de la relaxation

$|R| = 1, B = 4, \mathcal{T} = [0, 30]$



i	p_i	b_i
1	3	2
2	5	3
3	1	3
4	3	1
5	2	1
6	4	2
7	5	3
8	6	1
9	4	1
10	4	1



Borne inférieure = 17.14 (18) meilleur que la borne triviale

Variantes : variables en escalier, on/off, ...

- Beaucoup de variantes sont présentées comme des «nouvelles» formulations
- En fait la plupart de ces formulations sont équivalentes (elles ont toutes la même projection $\tilde{\mathcal{S}}_T$ et s'obtiennent les unes à partir des autres par des transformations non singulières.
  - **Exemple 1** : variable ξ_{it} en escalier ($\xi_{it} \Leftrightarrow S_i \leq t$)
    $\xi_{it} = \sum_{\tau=0}^{t} x_{it}$ et réciproquement $x_{it} = \xi_{it} - \xi_{it-1}$
  - **Exemple 2** : variable on/off μ_{it} ($\mu_{it} = 1 \Leftrightarrow t \in [S_i, S_i + p_i[$)
    $\mu_{it} = \sum_{\tau=t-p_i+1}^{t} x_{i\tau}$ et, réciproquement,
    $x_{it} = \sum_{k=0}^{\lfloor t/p_i \rfloor} \mu_{i,t-kp_i} - \sum_{k=0}^{\lfloor (t-1)/p_i \rfloor} \mu_{i,t-kp_i-1}$
- Dans [A. 2017], les formulations proposées par [Klein 2000], [Bianco and Caramia 2013] sont montrées équivalentes à ou moins fortes que (DDT)

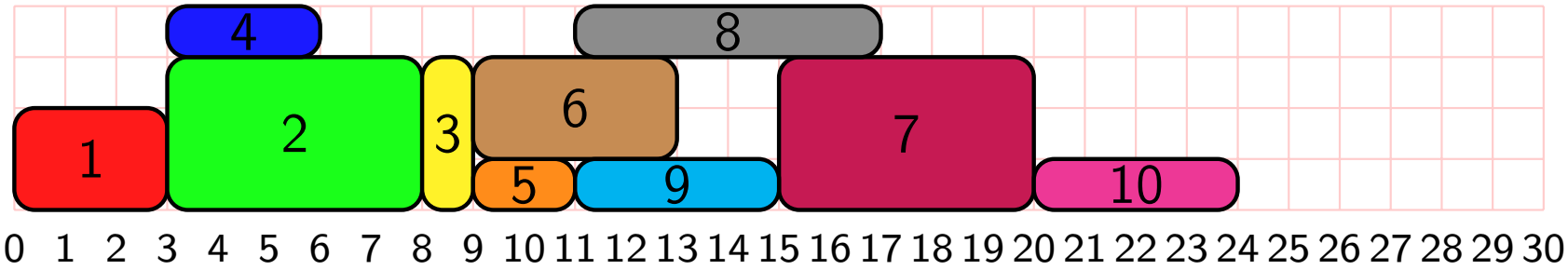
Inégalités valides

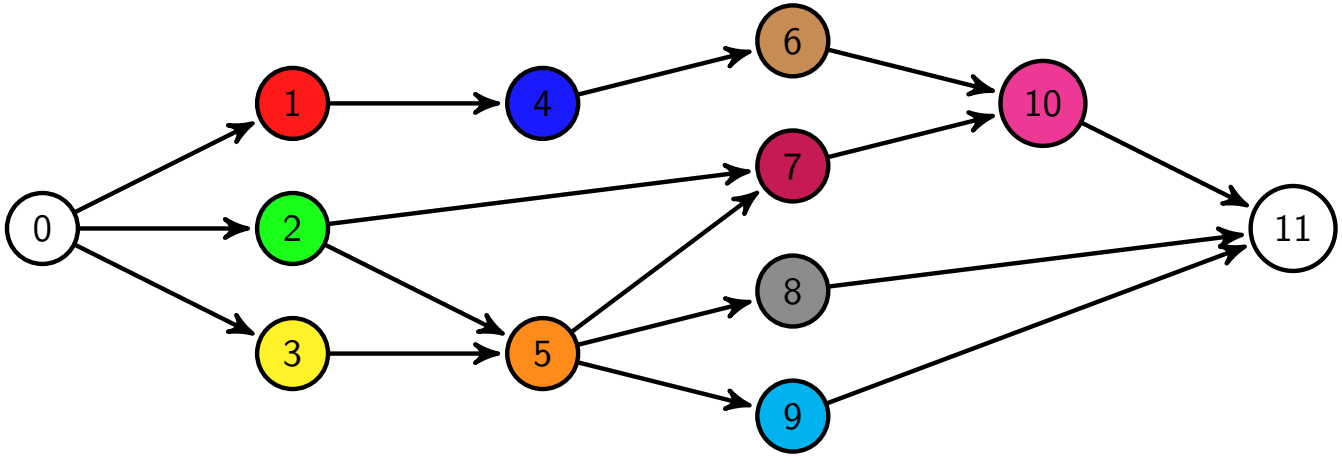
- Inégalités valides basées sur les ensembles critiques [Hardin *et al* 2008]
  - Basic inequality: $\sum_{i \in A} \sum_{s=t-p_i+1}^{t} x_{is} \leq |F| - 1, \quad \forall F \in \mathcal{F}$
  - Famille plus générale, extension aux intervalles de longueur v

$$\sum_{i \in F \setminus \{j\}} \sum_{s=t-p_i+1+v}^{t} x_{is} + \sum_{s=t-p_j+1}^{t+v} x_{js} \leq |F| - 1 \quad \forall F \in \mathcal{F}$$

  - Procédure de lifting et séparation heuristique
- Autres inégalités [Christofides et al. 1987, de Sousa and Wolsey 1997, Cavalcante et al. 2001, Baptiste and Demassey 2004, Demassey *et al* 2005]

Solution : séquence d'antichaines réalisables





Ensemble d'antichaines respectant les contraintes de ressources

$\{\{1\},\{1,5\},\{1,8\},\{1,9\},\{1,8,9\},\{2\},\{2,4\},\{3\},\{3,4\},\{4\},\{4,5\},\{4,7\},\{4,8\},\{4,9\},$
$\{4,8,9\},\{5\},\{5,6\},\{6\},\{6,8\},\{6,9\},\{6,8,9\},\{7\},\{7,8\},\{7,9\},\{8\},\{8,9\},$
$\{8,10\},\{8,9,10\},\{9\},\{9,10\},\{10\}\}$

Solution représentée:

$\{1\} \prec \{2,4\} \prec \{2\} \prec \{3\} \prec \{5,6\} \prec \{6,8,9\} \prec \{8,9\} \prec \{7,8\} \prec \{7\} \prec \{10\}$

Formulation étendue basée sur les antichaînes

Soit $\mathcal{P}$ l'ensemble des antichaînes

- $y_{Pt} = 1 \Leftrightarrow$ l'antichaine P est en cours d'exécution à la période t.
- Formulation obtenue de (DDT) par décomposition de Dantzig-Wolffe, en remplaçant les contraintes de ressources par :

$$\sum_{P \in \mathcal{P}_i} \sum_{t \in T} y_{Pt} = p_i \quad i \in A,\ p_i \geq 1$$

$$\sum_{P \in \overline{\mathcal{P}}} y_{Pt} \leq 1 \quad t \in T$$

$$x_i^t - \sum_{P \in \mathcal{P}_i} y_{Pt} - \sum_{P \in \mathcal{P}_i} y_{P,t-1} \geq 0 \quad i \in A;\ t \in T$$

$$y_{At} \in \{0,1\} \quad P \in \mathcal{P};\ t \in \cap_{i \in P}\{ES_i, \ldots, LS_i\}$$

où $\mathcal{P}_i \subseteq \mathcal{P}$ est l'ensemble des antichaînes qui contiennent la tâche i.
[Mingozzi *et al* 1998]

Bornes inférieures basées sur la formulation par antichaînes

Couplée à la propagation de contraintes qui permet de réduire les fenêtres de temps, cette formulation donne les meilleures bornes basées sur la PLNE pour le RCPSP.

- Weighted Node Packing basée sur le dual de la relaxation préemptive [Mingozzi *et al.* 1998]
- Borne destructive basée sur la propagation de contraintes et la relaxation préemptive de Mingozzi et al. résolue par génération de colonnes [Brucker and Knust 2000, Demassey *et al* 2004, Baptiste and Demassey 2004]
- Relaxation préemptive résolue par Branch&Price. [Moukrim *et al.* 2013]

Comparaison PLNE/MILP sur un problème industriel partiellement préemptif

- Problème d'ordonnancement d'activités du Laboratoire LECA/STAR du CEA (analyse de combustible nucléaire)
- Opérateurs multi-compétences
- Horizon d'une semaine
- Activités préemptives, non préemptives et partiellement préemptives
- Comparaison PLNE indexée par le temps vs CP Optimizer [Polo et al 2020]

Table 4. Distribution of preemption types per set of instances.

	Set A1	Set B1	Set C1	Set D1
Non-preemptive	10%	10%	80%	33.3%
Partially preemptive	10%	80%	10%	33.3%
Preemptive	80%	10%	10%	33.3%

Table 6. Results of MILP and CP models after 10 min of computation using warm start

	MILP			CP		
	Number of instances solved to optimality	Average time to optimality	Average gap	Number of instances solved to optimality	Average time to optimality	Average gap
Set A1	46	87.39 s	0.05%	39	67.17 s	0.18%
Set B1	15	154.12 s	2.69%	40	88.01 s	0.15%
Set C1	0	-	9.45%	41	108.73 s	0.39%
Set D1	19	216.12 s	1.99%	40	76.14 s	0.21%
All	80	130.48 s	3.55%	160	85;27 s	0.23%

subsection in toc subsection in toc subsection in toc

Recherche arborescente dirigée par les conflits

- La meilleure méthode exacte pour le RCPSP est longtemps restée celle de [Demeulemeester & Herroelen (1997)]
  - Borne inférieure «weighted node packing»
  - Branchement sur les alternatives minimales de résolution des conflits.
  - Mémorisation des «cutsets» (ensemble d'activités non ordonnancées dont tous les prédécesseurs sont ordonnancées) ayant amené à un échec et comparaison du nœud courant à la base de cutsets.
- Les meilleurs méthodes actuelles exploitent ce type recherche dirigée par les conflits de manière systématique et optimisée en utilisant le principe des solveurs SAT.

Satisfaisabilité booléenne (SAT)

Problem

- Variables booléennes (atome)
- Formules de logique propositionnelle (CNF)
- Littéraux: $a, \overline{a}$
- Clauses: $(\overline{a} \vee \overline{f} \vee g),\ (\overline{a} \vee \overline{f} \vee g),\ (\overline{a} \vee \overline{b}),\ (b \vee \overline{c} \vee g)$
- Solution: affectation des atomes satisfaisant toutes les clauses

Algorithmes

- Recherche locale (GSAT, WalkSat,...)
- Propagation
- DPLL: Recherche arborescente + propagation unitaire
- CDCL: Conflict Driven Clause learning

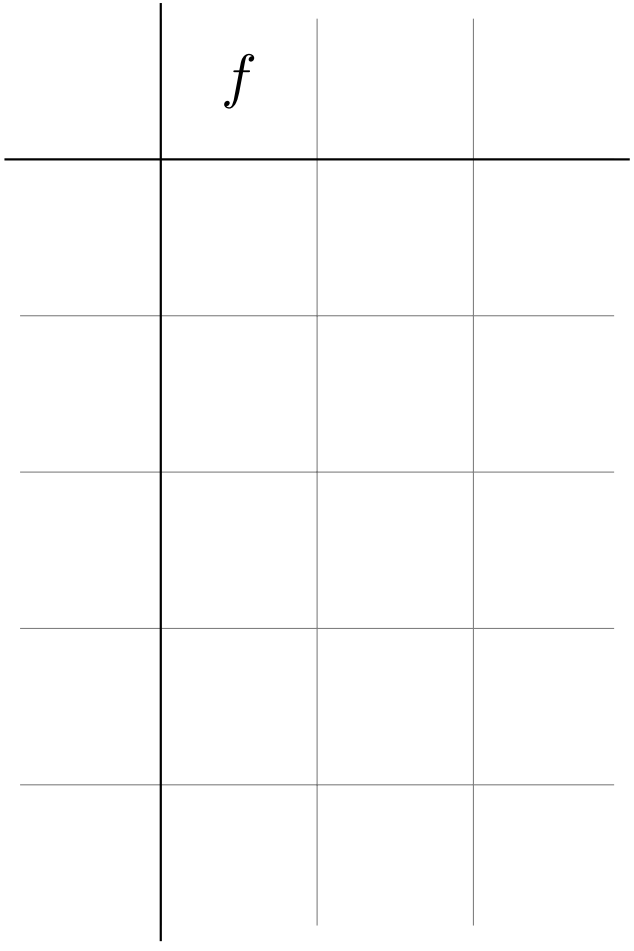
Apprentissage de clauses dirigé par les conflits (CDCL)

issu de DPLL

- apprentissage de clause (solveurs `GRASP puis Chaff`)
  - `Première compétition SAT-Solver 2002`

- Prendre des décisions
  - Propagation unitaire: si a doit être vrai, alors $\overline{a}$ ne peut pas satisfaire de clause
  - $\overline{a} \vee b \vee \overline{c}$ devient $b \vee \overline{c}$
    - continue jusqu'à atteindre un point fixe
- Jusqu'à détecter un conflit (impasse)
  - Extraire une clause apprise
  - Effectuer un backump et propager la clause apprise
- Heuristiques de branchement adaptative (pondération des littéraux en conflit)
- Et aussi: redémarages, simplification de la base de clause, oubli de clauses, etc.

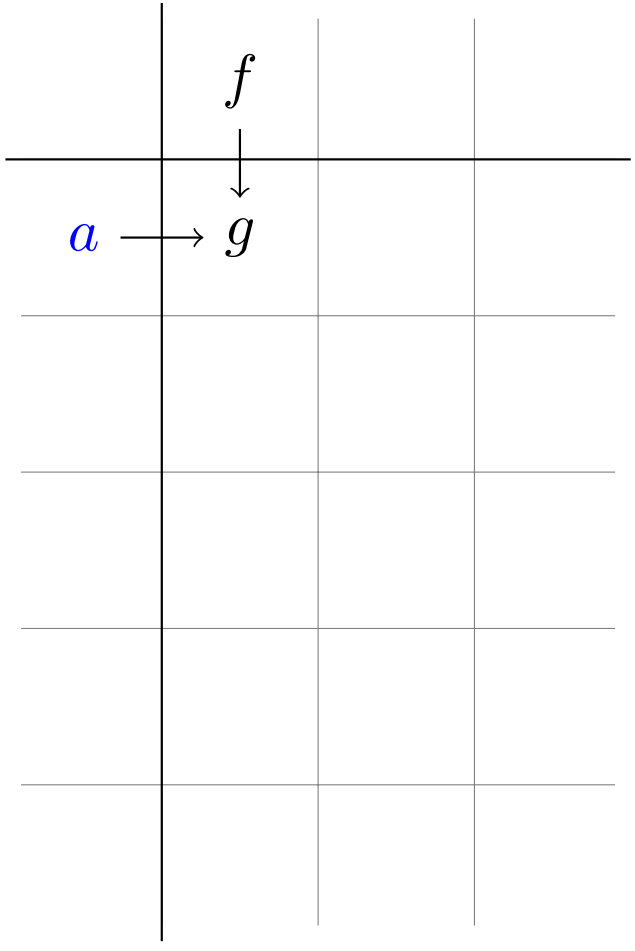
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

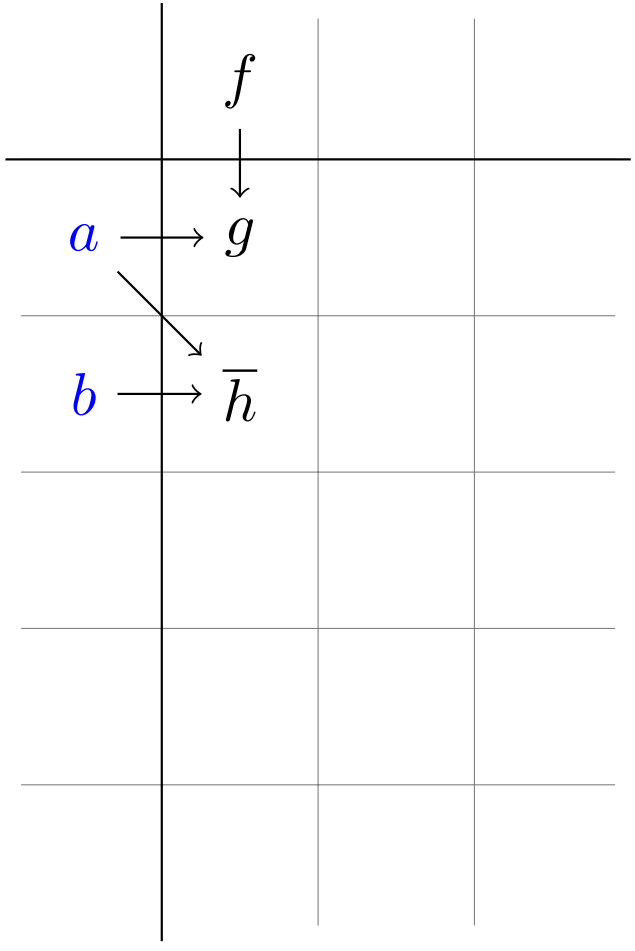
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

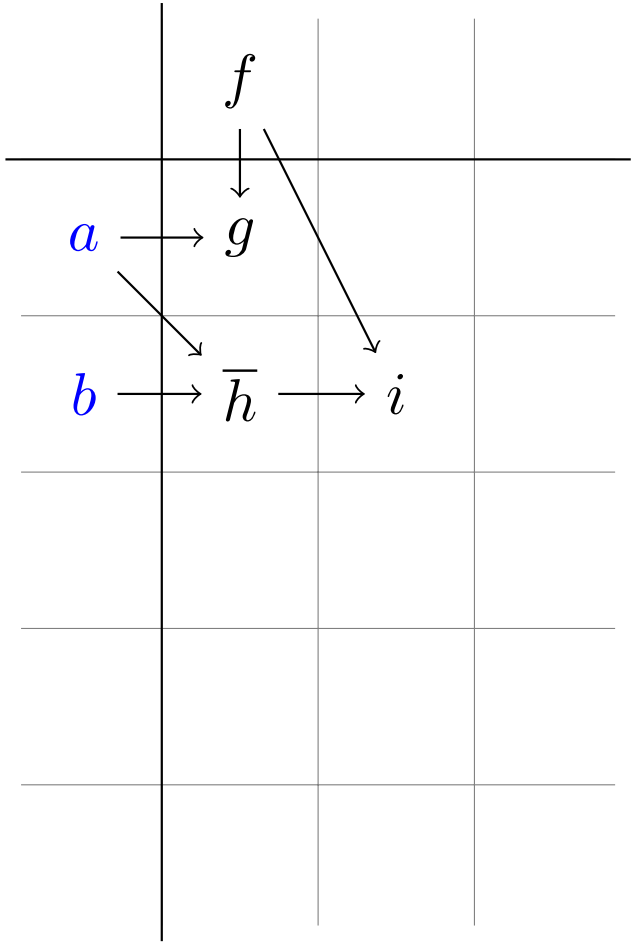
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

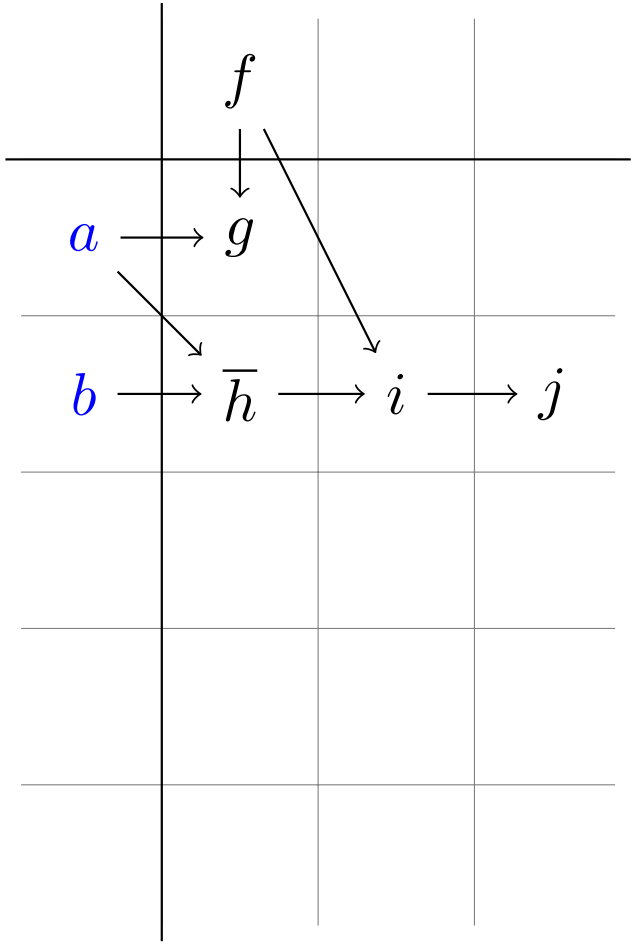
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

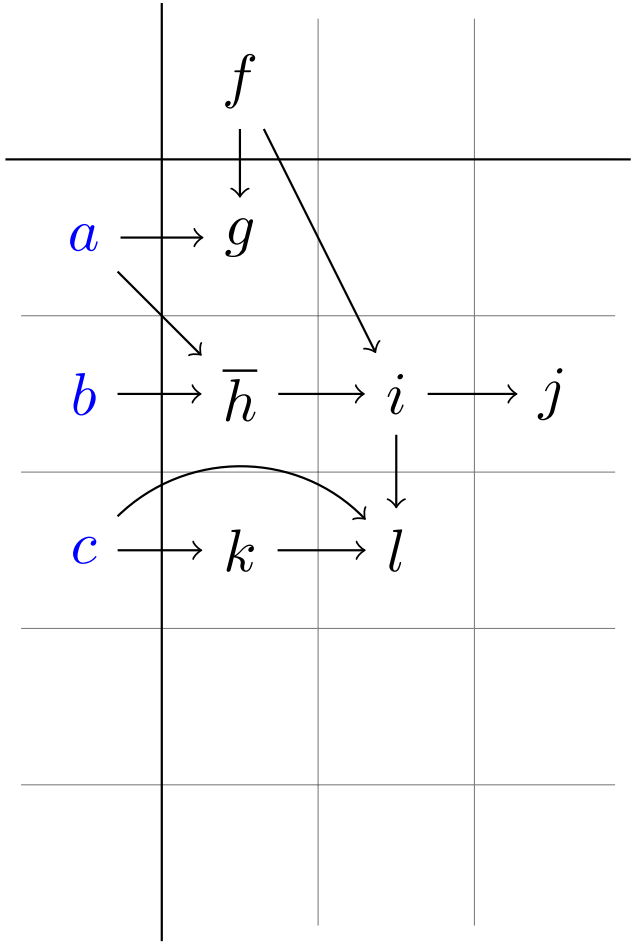
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

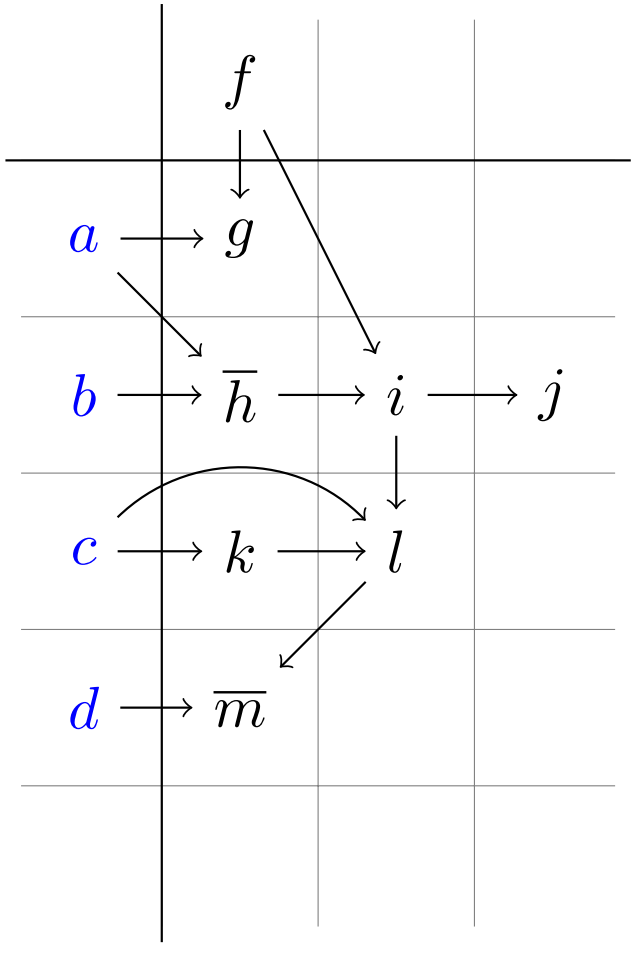
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

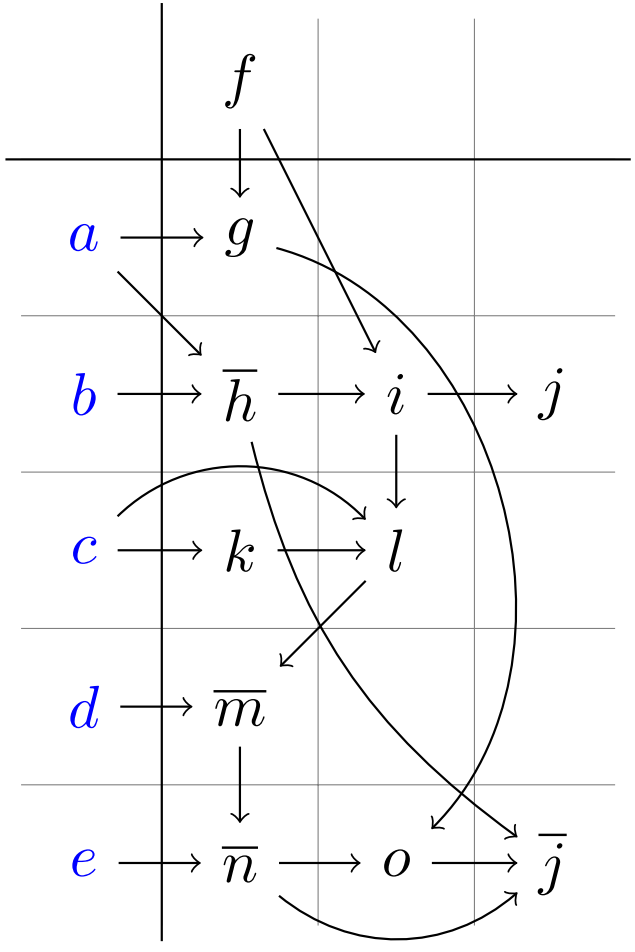
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

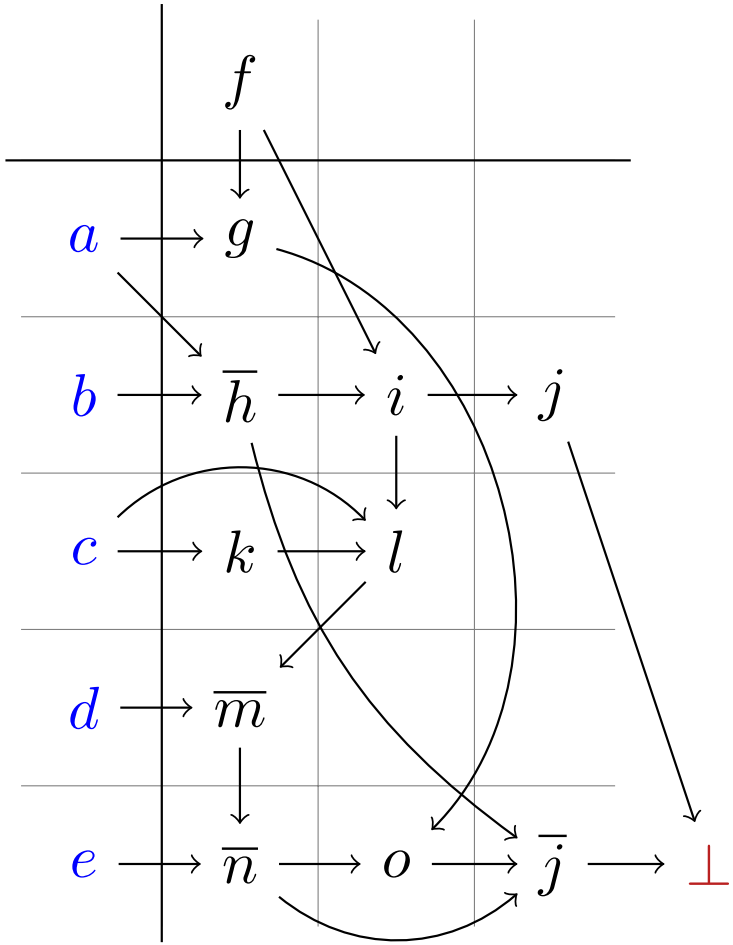
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

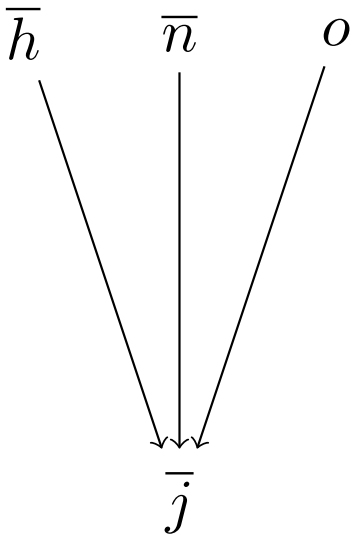
CDCL: Exemple



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

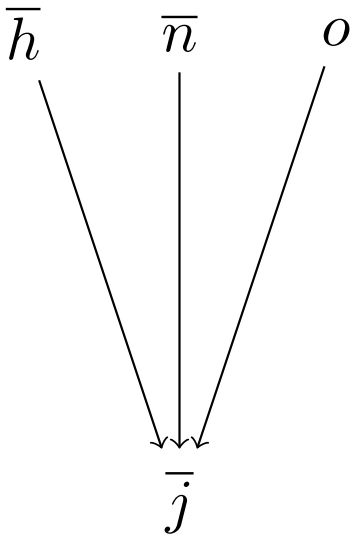
$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$\boxed{h \vee \overline{o} \vee \overline{j} \vee n}$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

CDCL: Example



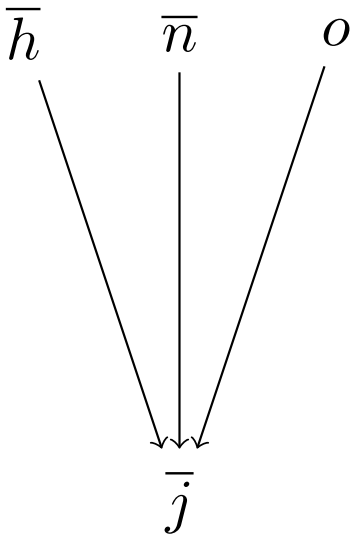
$$(h \vee \overline{o} \vee \overline{j} \vee n)$$

CDCL: Example



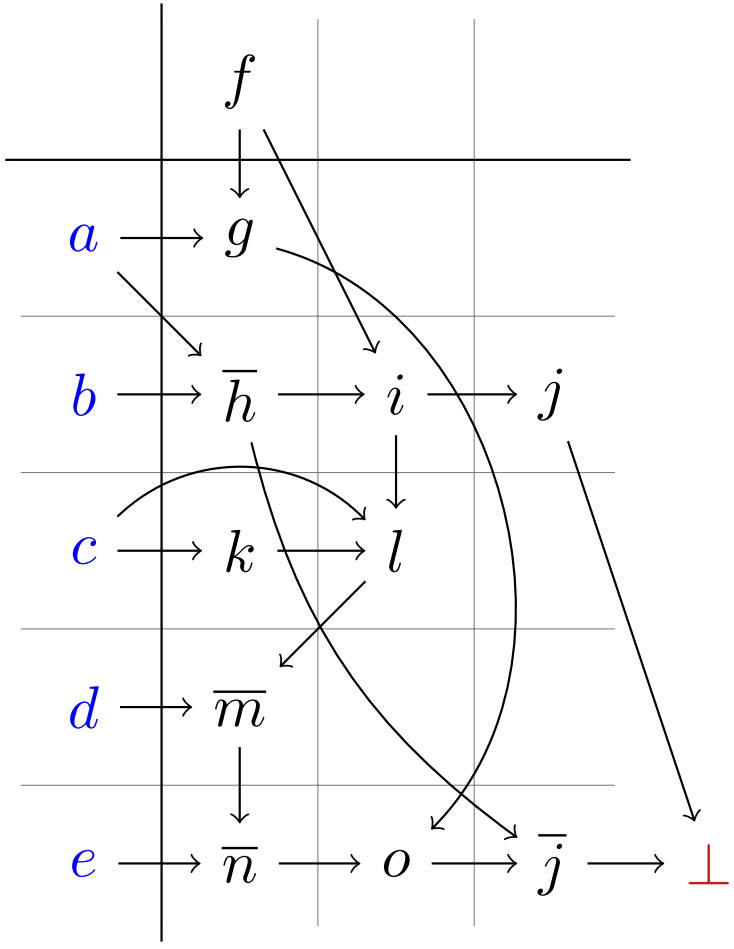
$$(h \vee \overline{o} \vee \overline{j} \vee n)$$

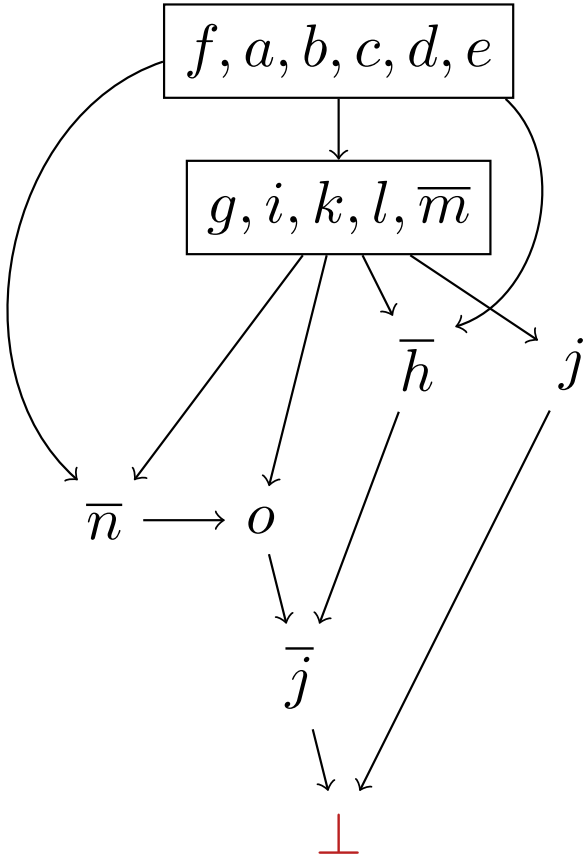
CDCL: Example



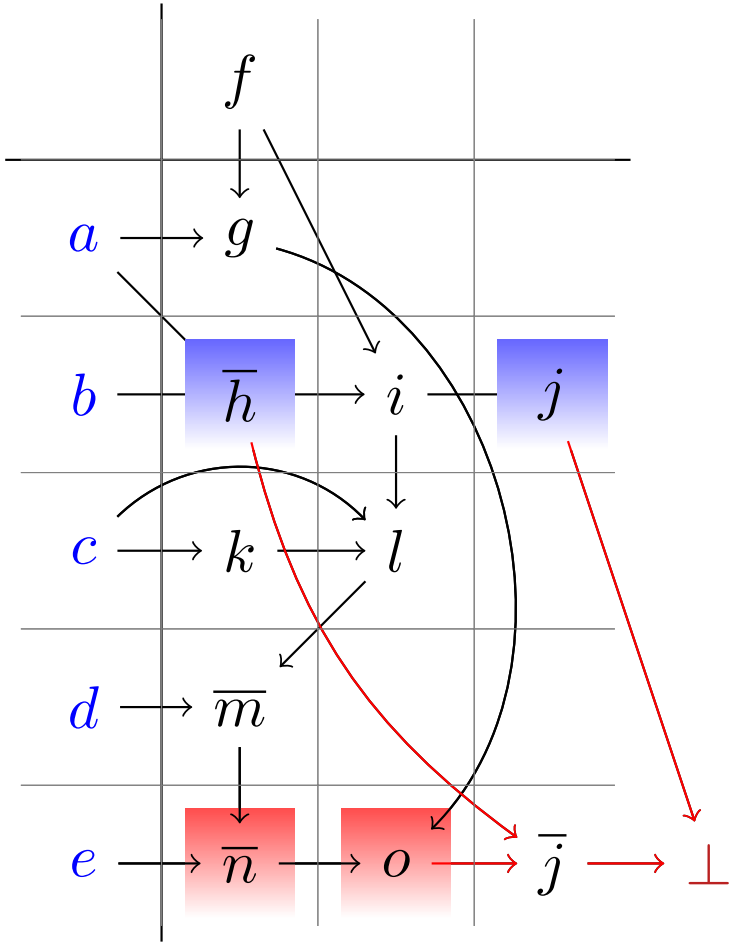
$$(h \vee \overline{o} \vee \overline{j} \vee n)$$
$$\equiv$$
$$(\overline{h} \wedge o \wedge \overline{n}) \rightarrow \overline{j}$$

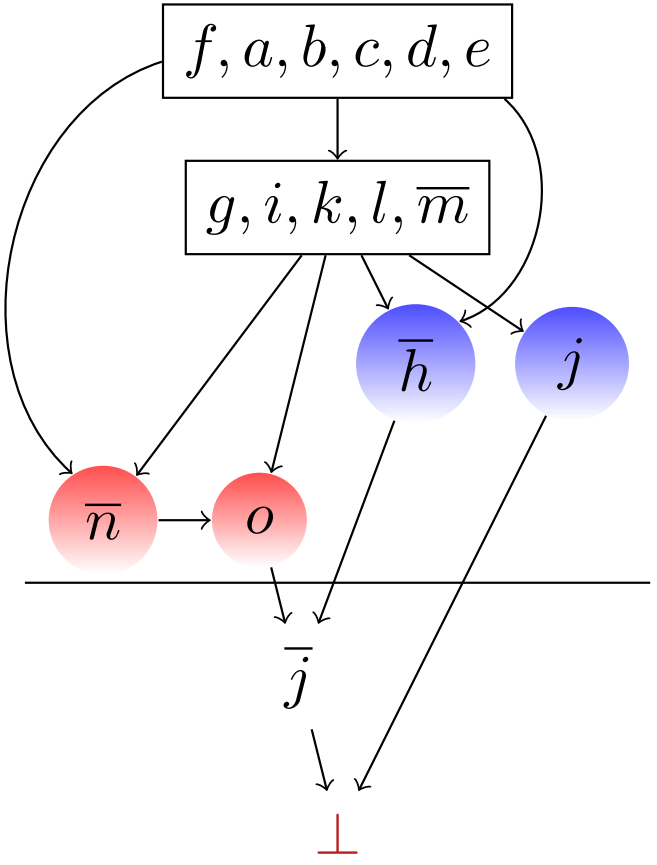
CDCL: Example



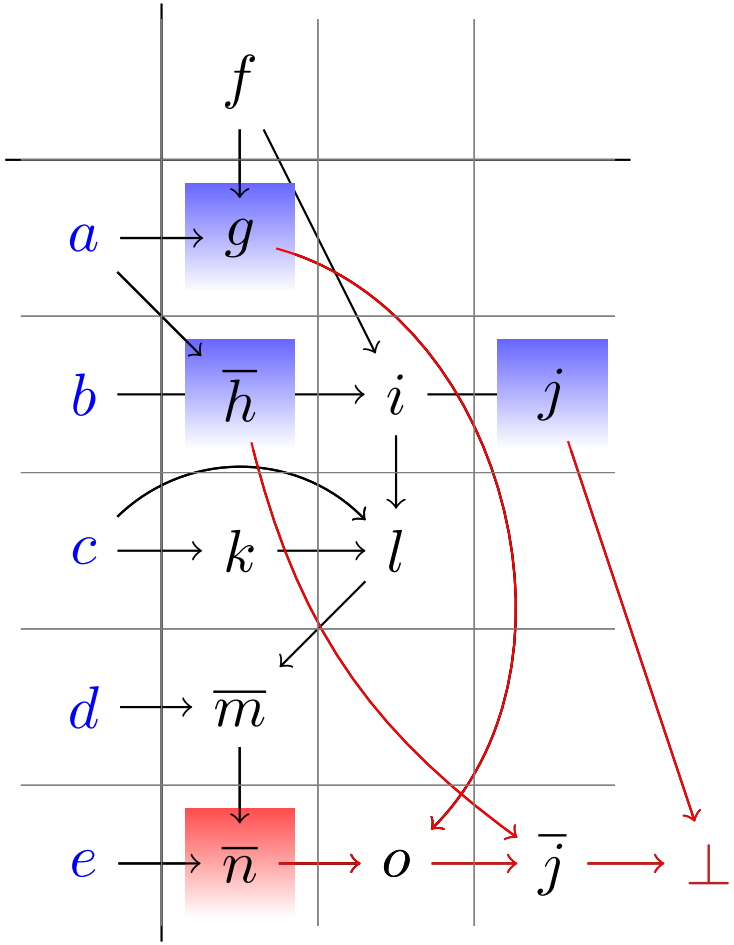


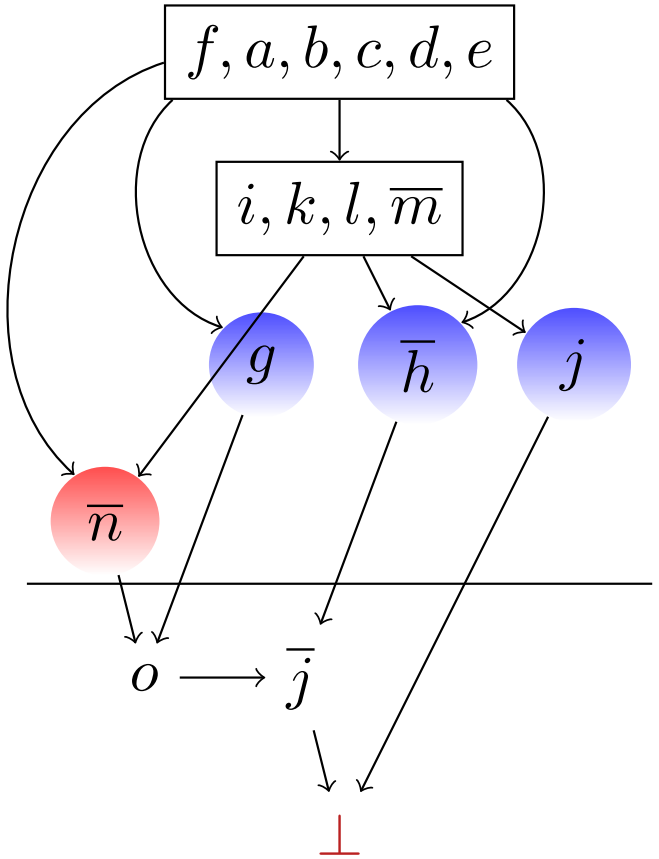
CDCL: Example



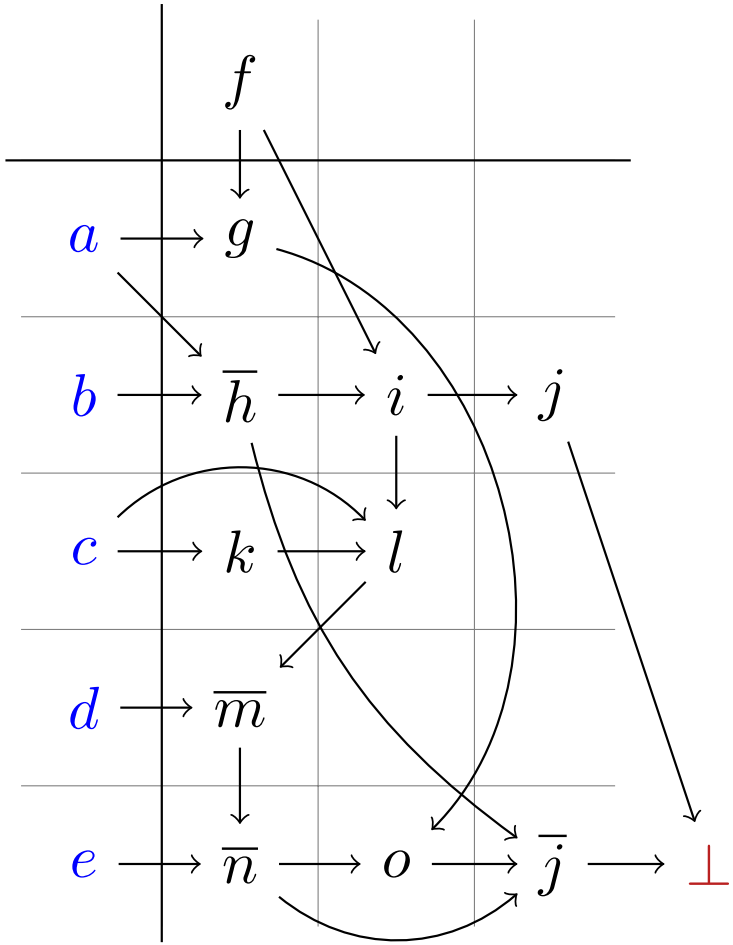


CDCL: Example





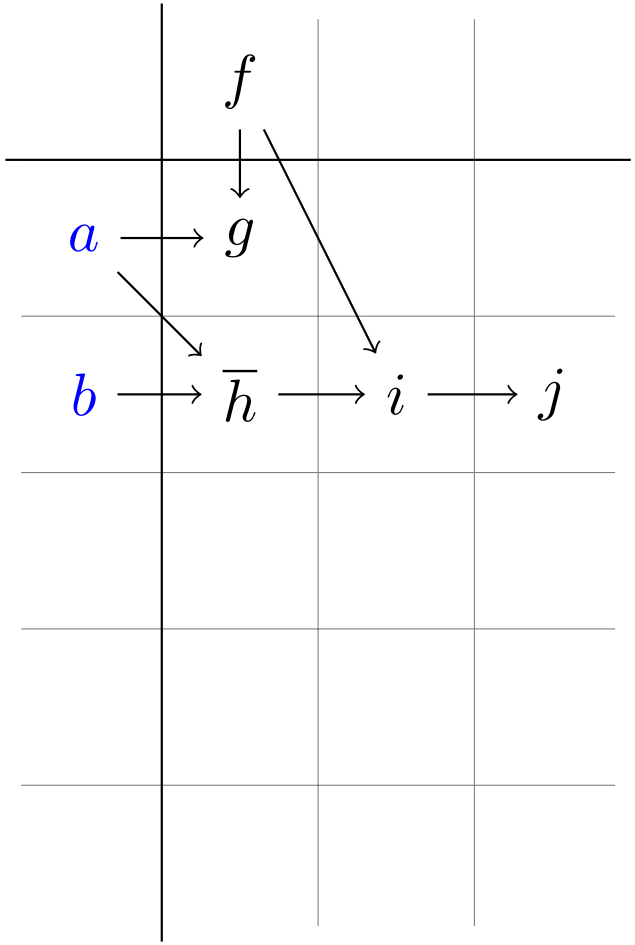
CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

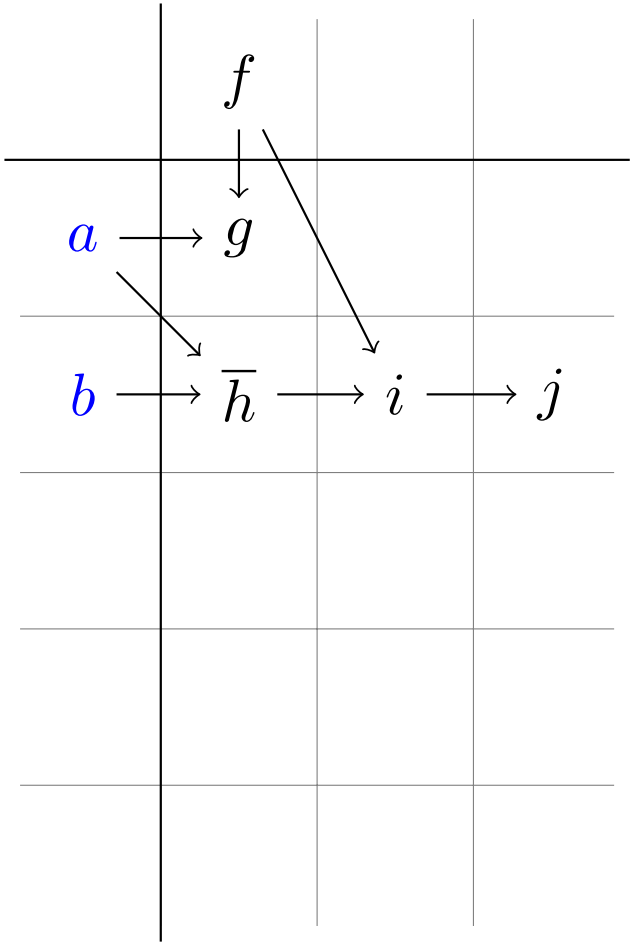
CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$

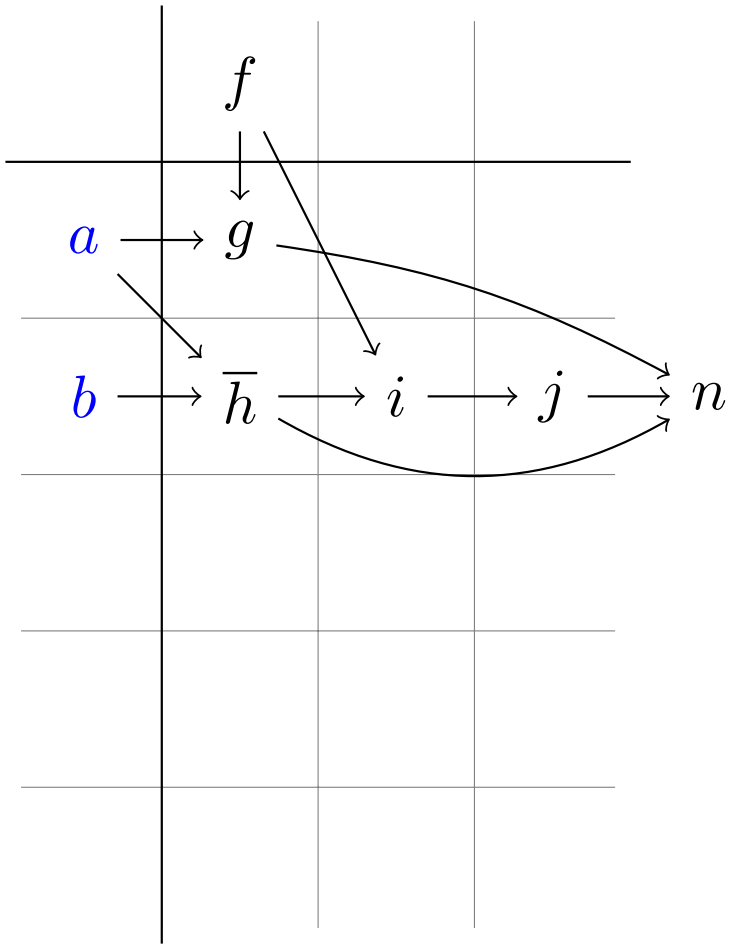
CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$
$\boxed{\overline{g} \vee h \vee \overline{j} \vee n}$

CDCL: Example



$\overline{a} \vee \overline{f} \vee g$
$\overline{a} \vee \overline{b} \vee \overline{h}$
$a \vee c$
$a \vee \overline{i} \vee \overline{l}$
$a \vee \overline{k} \vee \overline{j}$
$b \vee d$
$b \vee g \vee \overline{n}$
$b \vee \overline{f} \vee n \vee k$
$\overline{c} \vee k$
$\overline{c} \vee \overline{k} \vee \overline{i} \vee l$

$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$h \vee \overline{o} \vee \overline{j} \vee n$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$
$\overline{f} \vee h \vee i$
$\boxed{\overline{g} \vee h \vee \overline{j} \vee n}$

Encodage SAT d'un problème d'ordonnancement

- Encodage de variables booléennes (par exemple séquencement), de variables entières (par exemples dates de début)
- Exemple : Order Encoding
  - Un atome i_v pour chaque paire $(S_i, v \in [ES_i, LS_i])$
    - $i_v \Leftrightarrow x_i \leq v$

$$\begin{array}{ll} S_i = 1: & 1111 \\ S_i = 2: & 0111 \\ S_i = 3: & 0011 \\ S_i = 4: & 0001 \end{array}$$

  - propagation des bornes
    - Si $x_i \leq v$ alors $x_i \leq v+1$
    - $\bigwedge_{v \in [ES_i, LS_i]} \overline{i_v} \vee i_{v+1}$ (n-1 clauses binaires)
  - Exemple : encodage des contraintes de précédence :

$S_j \geq S_i + p_i \Leftrightarrow$

$S_j \leq t \Rightarrow S_i \leq t - p_i, \forall t \in D_j, t - p_i \in D_j \Leftrightarrow$

$j_t \Rightarrow i_{t-p_i}, \forall t \in D_j, t - p_i \in D_j \Leftrightarrow$

$\overline{j_t} \vee i_{t-p_i}, \forall t \in D_j, t - p_i \in D_j$

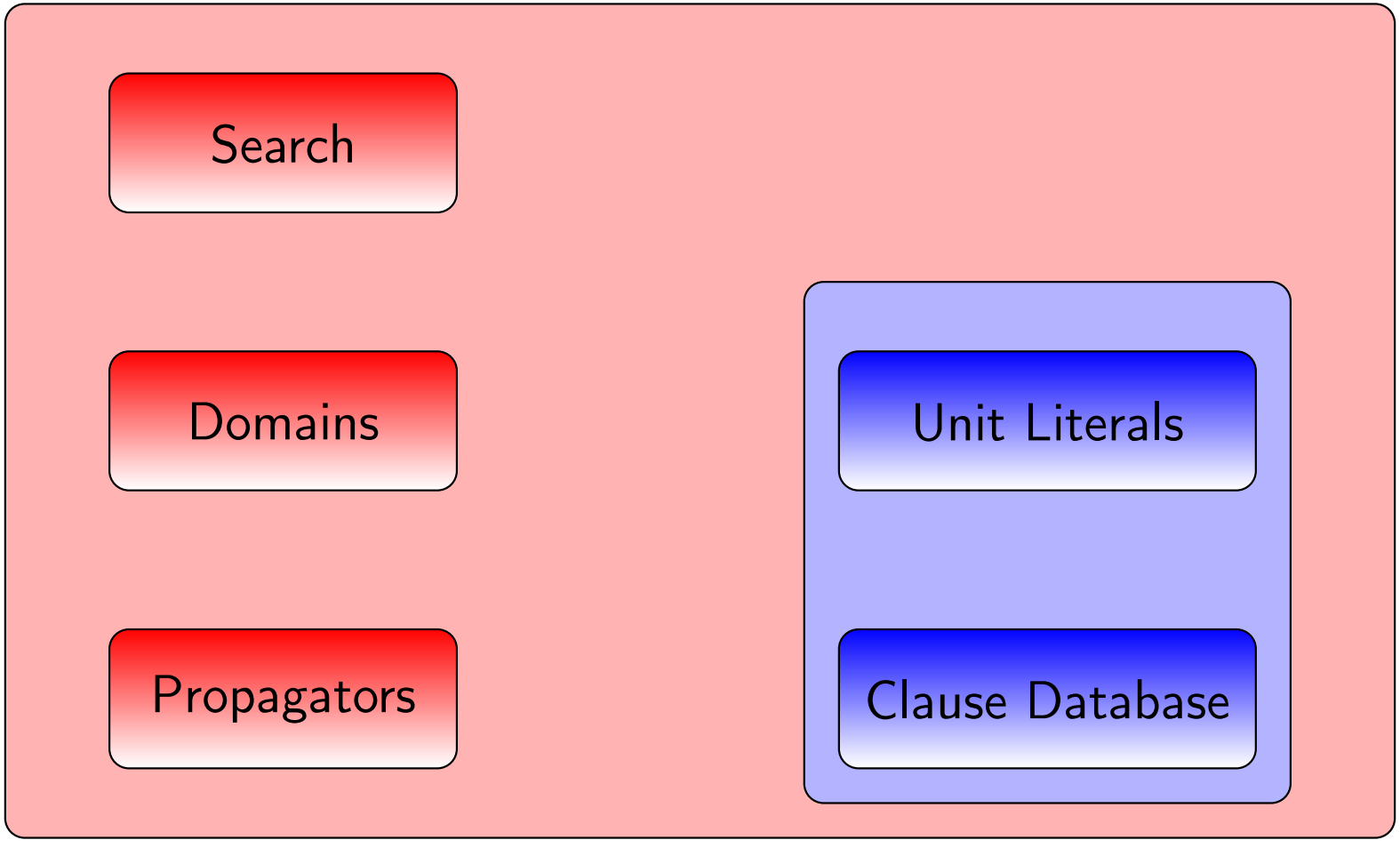
Résultats sur les problèmes d'ordonnancement

Resultat de l'encodage SAT

- Encodage plus efficace : mélange d'Order Encoding et de "Log Encoding"
- [Tamura, Tanjo & Banbara 2012] : solveur (Azucar) capable de résoudre toutes les instances ouvertes d'open shop.
- Encodage SAT $\rightarrow$ parfois une méthode efficace !

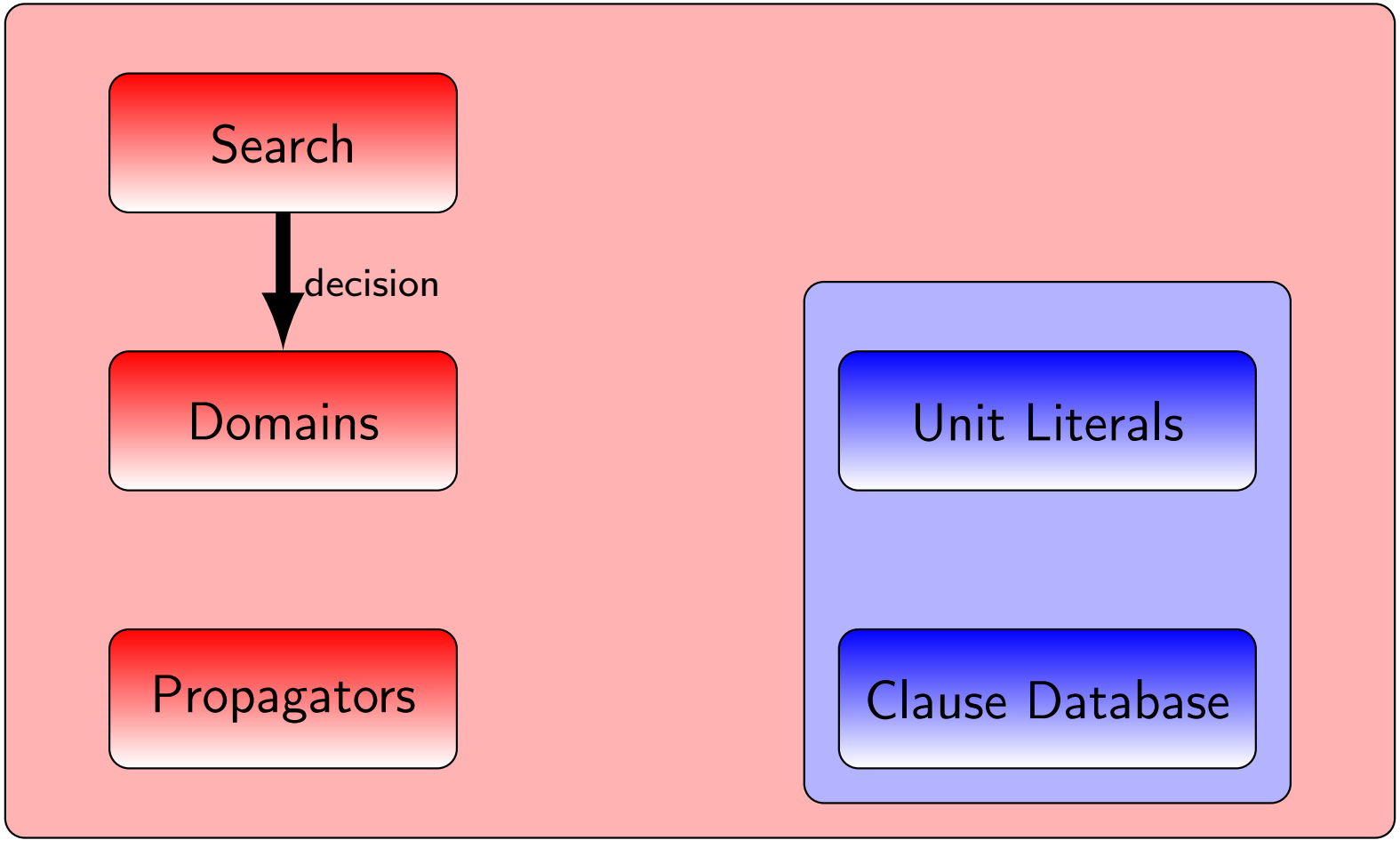
La génération de clauses retardée

Architecture



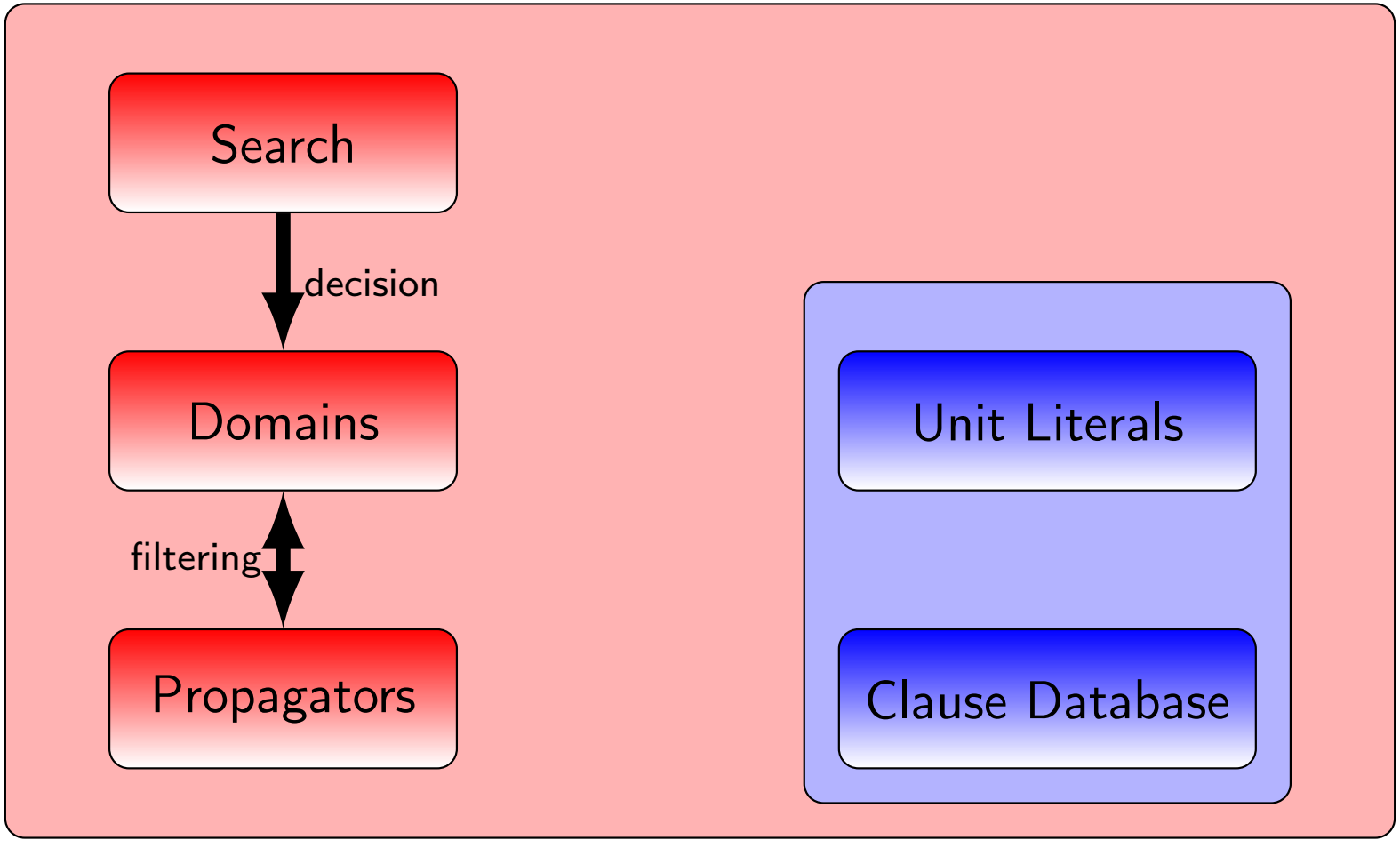
La génération de clauses retardée

Architecture



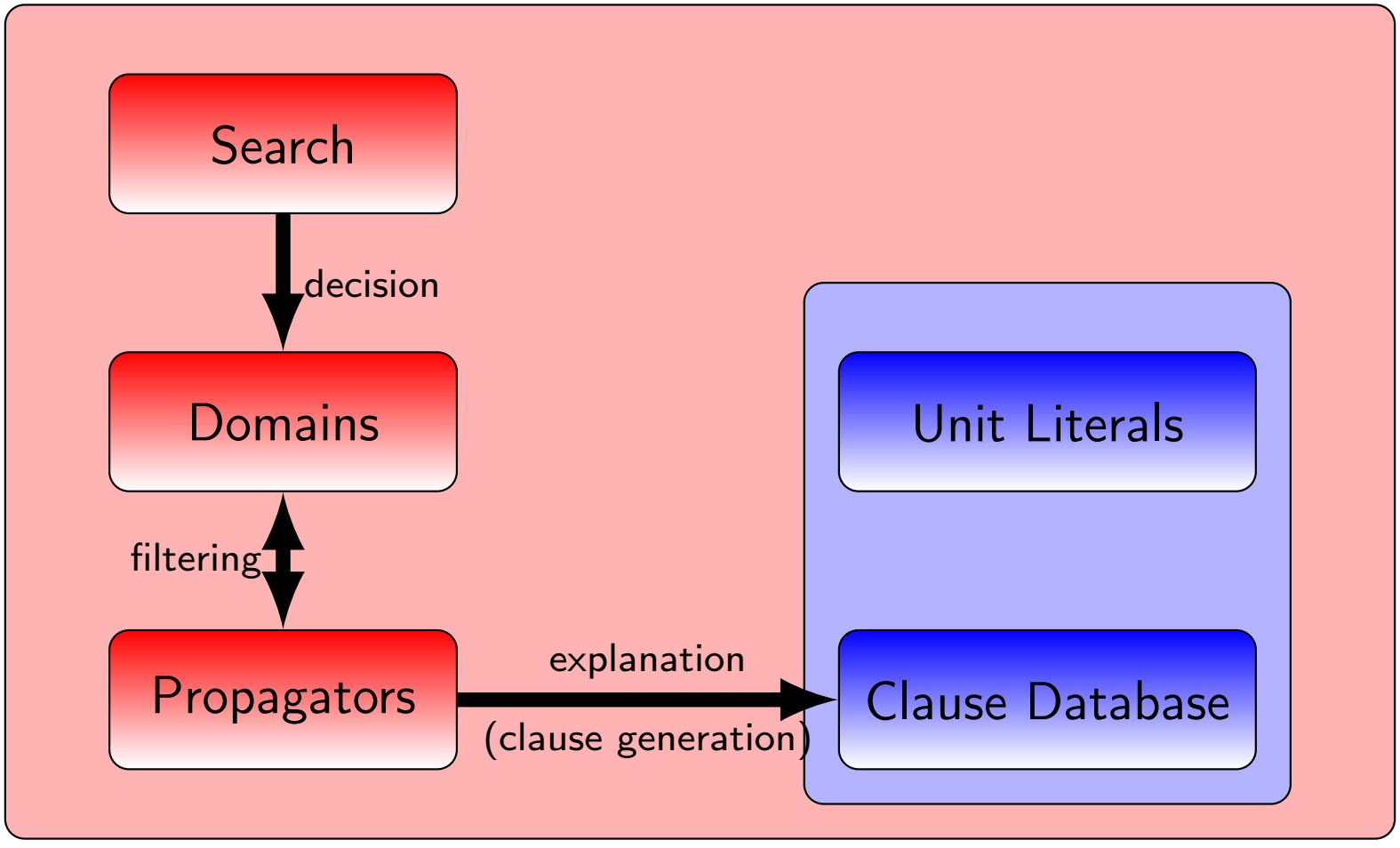
La génération de clauses retardée

Architecture



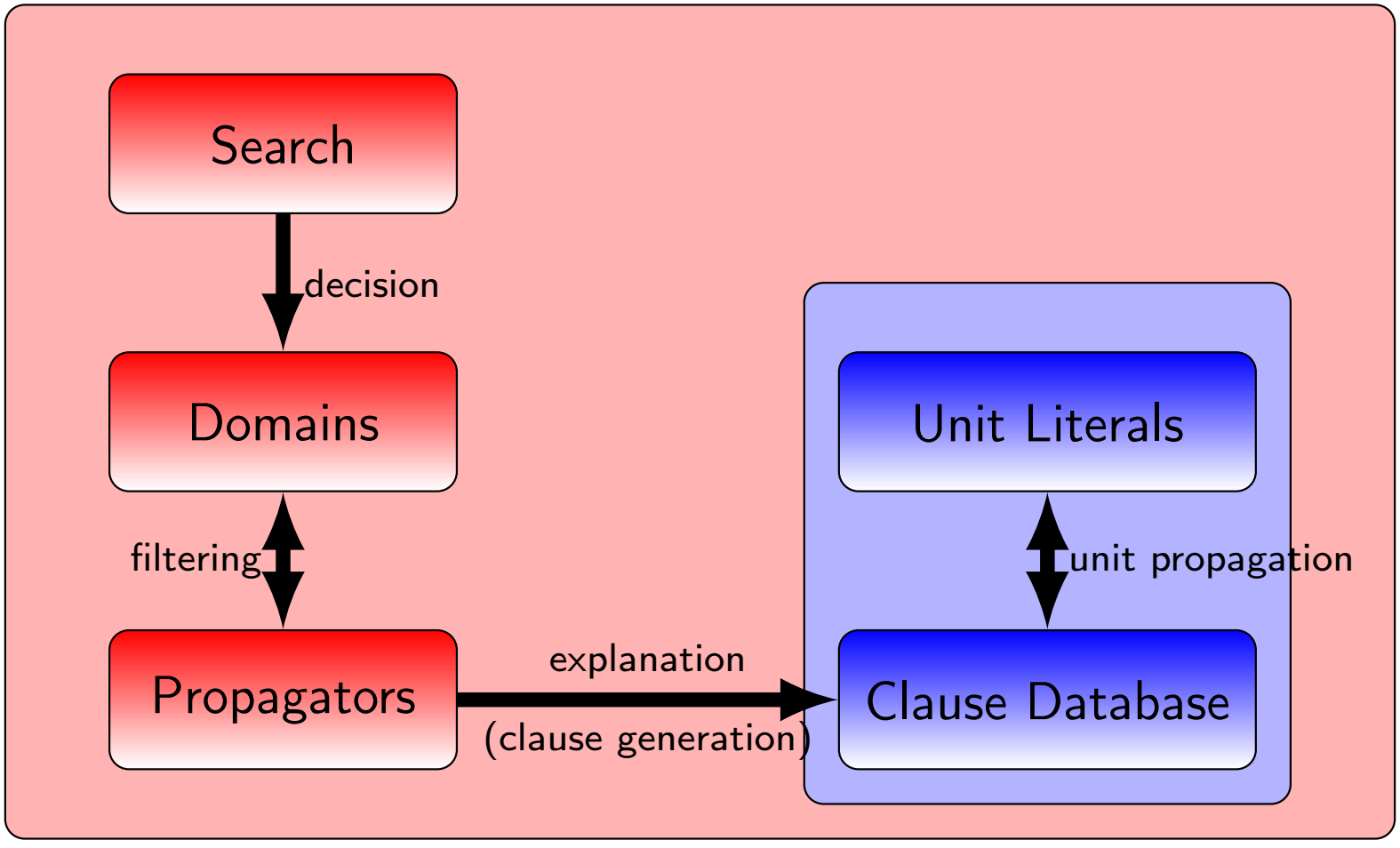
La génération de clauses retardée

Architecture



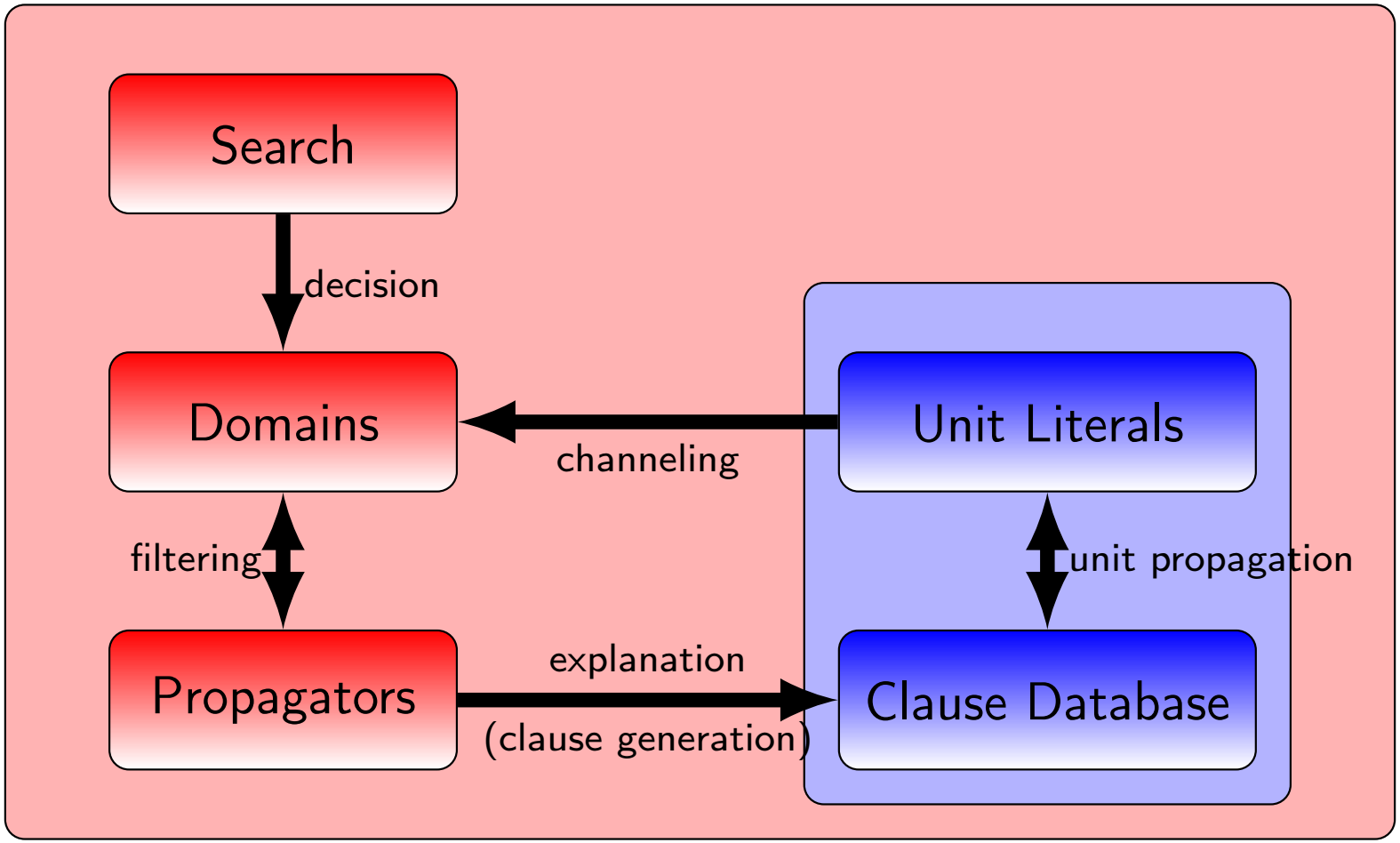
La génération de clauses retardée

Architecture



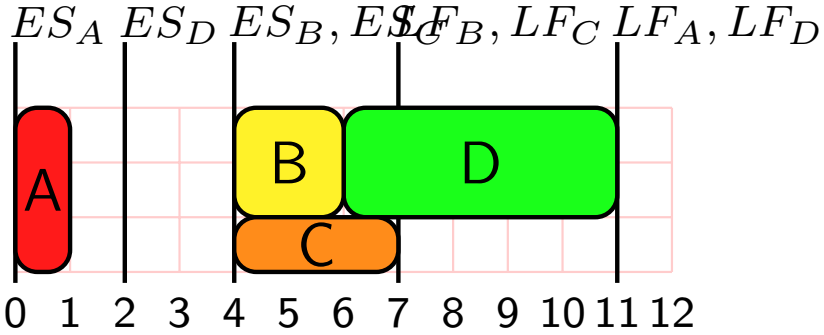
La génération de clauses retardée

Architecture



Application à l'ordonnancement

Génération retardée de clauses [Schutt et al. 2009,2011,2013] basée sur le concept de no-good généralisé [Katsirelos & Bacchus 2005]



- On cherche l'explication la plus générale possible : explication basée sur le raisonnement énergétique.
- Exemple :
  $[\![S_D \geq 1]\!] \wedge [\![S_B \geq 4]\!] \wedge [\![S_B \leq 5]\!] \wedge [\![S_C = 4]\!] \implies [\![S_D \geq 6]\!]$
- Autres approches de recherche arborescente guidée par les échecs : Conflict Ordering Search [Gay et al 2015], Failure Directed Search [Vilím et al 2015], (RQ: Pour le job-shop 2 bornes améliorées sur FDS [Siala et al. 2015])

Quelques comparaisons: bornes inférieures

inst	BD04	LCG12	%DDT	PFS	FDS15
9_1	82	85	-2.35%		
9_3	91	99	-9.09%		
9_5	78	81	-3.70%		82
9_6	100	105	-4.76%		
9_7	101	105	-2.86%		
9_8	89	95	-7.37%		
9_9	92	99	-7.07%		
9_10	86	90	-3.33%		91
13_1	104	105	-1.90%	107	
13_2	101	103	-1.94%		
13_3	82	84	-1.19%		
13_4	97	98	-3.06%		
13_5	91	92	-1.09%		93
13_6	90	91	-1.10%		
13_7	80	83	-3.61%		
13_8	112	115	-3.48%		
13_9	95	97	-2.06%		
13_10	112	114	-0.88%		
25_2	91	95	-5.26%		
25_4	98	106	-8.49%		
25_6	103	105	-4.76%		
25_7	83	88	-6.82%		
25_8	90	95	-5.26%		96
25_10	99	107	-6.54%		

inst	BD04	LCG12	%DDT	PFS13	FDS
29_1	96	98	-3.06%		
29_2	123	123	-7.32%	127	
29_3	115	114	-1.75%	118	115
29_4	126	126	-7.14%	130	
29_5	102	102	-3.92%	105	
29_6	143	144	-9.03%	146	145
29_7	114	117	-4.27%		
29_8	96	98	-2.04%		
29_9	104	105	-4.76%	107	112
29_10	111	111	-1.80%		
30_2	67	69	-1.45%		
41_3	88	90	-4.44%		
41_5	105	109	-7.34%		
41_10	104	108	-2.78%		
45_1	89	90	-4.44%	91	
45_2	134	134	-11.94%	138	
45_3	132	133	-6.02%	138	
45_4	101	101	-4.95%	103	
45_5	99	99	-3.03%	101	100
45_6	133	132	-21.21%	137	133
45_7	113	113	-5.31%	117	
45_8	119	119	-5.04%	123	
45_9	115	114	-5.26%	119	
45_10	103	102	-3.92%	107	105

BD04 [Demassey&Baptiste 04] CP&CG ; LCG12: [Schutt *et al* 13] (Lazy clause generation)
PFS13: [Moukrim *et al* 13] Preemptive feasible subset formulation solved by B&P

FDS: [Vilím *et al* 15] Failure directed search

Quelques comparaisons: résolution exacte

Instances	Formulations	%Integer	%Opt	%Gap	%ΔCPM	Time Opt (s)
KSD30	DDT	91	82	0.47	8.91	10.45
	DT	86	78	0.55	6.74	12.76
	FCT	67	62	0.16	3.76	22.66
	OOE_Prec	46	30	1.69	13.65	52.31
	OOE	33	24	1.22	7.00	112.62
	SEE	3.1	2.9	0.24	0.61	123.62
	MCS	-	**97**	0.00	11.48	7.39
PACK	DDT	95	76	1.08	199.02	63.39
	DT	85	55	0.49	203.58	48.24
	OOE_Prec	55	5	3.25	227.19	18.92
	OOE	49	9	2.89	231.29	61.78
	FCT	2	0	1.28	14.49	-
	SEE	0	0	-	-	-
	MCS	-	25	0.00	149.81	115.88
BL	DDT	100	100	0.00	32.40	13.68
	DT	100	100	0.00	32.40	37.93
	OOE_Prec	54	0	7.26	40.30	-
	OOE	49	0	7.90	41.65	-
	FCT	21	3	6.14	30.64	310.58
	SEE	8	0	12.81	29.96	-
	MCS	-	**100**	0.00	32.40	3.29
KSD15_d	OEE_Prec	99.8	86	0.00	10.02	6.49
	FCT	99	94	0.02	9.02	12.06
	OEE	99	83	0.01	10.14	4.68
	SEE	92	76	0.15	9.86	13.04
	DT	55	54	0.23	4.31	12.10
	DDT	1	1	0.00	2.63	3.34
	MCS	-	**100**	0.00	10.18	0.07
PACK_d	OEE	60	18	1.26	120.13	75.58
	OOE_Prec	60	14	1.62	117.56	54.35
	FCT	7	7	0.00	0.00	60.88
	SEE	4	4	0.00	0.00	215.08
	DT	0	0		-	-
	DDT	0	0		-	-
	MCS	-	**38**	0.00	50.59	72.34

- MCS [Laborie 2005] (MFS-based CP)
- LCG [Schutt *et al* 2013]

	KSD30	PACK	BL	KSD15_d	PACK_d
LCG	100	70.91	100	100	67.27
MCS	82	25	100	100	38
MIP	97 (DDT)	76 (DDT)	100 (DDT)	94 (FB)	18 (OOE)

- KSD30 instances fortement disjonctives
- PACK,BL instances fortement cumulative
- KSD15_d : durées modifiées
- PACK_d : durées modifiées

Par ordre d'apparition:

[Carlier 1982] J. Carlier, The one-machine sequencing problem, European Journal of Operational Research 11 (1), 42-47, 1982.
[Carlier & Pinson, 2004] J Carlier, E Pinson, Jackson's pseudo-preemptive schedule and cumulative scheduling problems, Discrete Applied Mathematics 145 (1), 80-94, 2004.
[Carlier et Pinson, 1989] J Carlier, É Pinson, An algorithm for solving the job-shop problem, Management science 35 (2), 164-176, 1989.
[Carlier et Pinson, 1990] J Carlier, É Pinson, A practical use of Jackson's preemptive schedule for solving the job shop problem, Annals of Operations Research 26, 269-287, 1990.
[Briand, 2010] Cyril Briand, Samia Ourari, Brahim Bouzouia: An efficient ILP formulation for the single machine scheduling problem. RAIRO Oper. Res. 44(1): 61-71 (2010)
[Li, 2011] Xinyu Li, An efficient memetic algorithm for solving the job shop scheduling problem, Computers & Industrial Engineering 60, 699-705, 2011
[Nuijten, 1994] W. Nuijten, Time and resource constrained scheduling : a constraint satisfaction approach, PhD Thesis, Technische Universiteit Eindhoven, 1994.
[Martin & Shmoys 96] Paul Martin, David B. Shmoys, A New Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem. IPCO 1996: 389-403.
[Carlier et Pinson, 1994] J Carlier, E Pinson, Adjustment of heads and tails for the job-shop problem, European Journal of Operational Research 78 (2), 146-161, 1994.
[Vilim 2004] Petr Vilím, O(n log n) Filtering Algorithms for Unary Resource Constraint. CPAIOR 2004: 335-347
[Brucker et al, 1994] Peter Brucker, Bernd Jurisch, Bernd Sievers, A Branch and Bound Algorithm for the Job-Shop Scheduling Problem. Discret. Appl. Math. 49(1-3): 107-127 (1994)

[Grimes and Hébrard 2015] Diarmuid Grimes, Emmanuel Hebrard Solving Variants of the Job Shop Scheduling Problem Through Conflict-Directed Search. INFORMS J. Comput. 27(2): 268-284 (2015)
[Peridy et Rivreau 2005] Laurent Péridy, David Rivreau, Local adjustments: A general algorithm. Eur. J. Oper. Res. 164(1): 24-38 (2005)
[Boussemart et al 2004] Frédéric Boussemart, Fred Hemery, Christophe Lecoutre, Lakhdar Sais: Boosting Systematic Search by Weighting Constraints. ECAI 2004: 146-150
[Lecoutre et al 2007] Christophe Lecoutre, Lakhdar Sais, Sébastien Tabary, Vincent Vidal: Nogood Recording from Restarts. IJCAI 2007: 131-136
[Adams et al 1988] Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. Management science, 34(3), 391-401.
[Hurink 2005] Johann Hurink, Scheduling (LNMB Master Course), `https://web-static.stern.nyu.edu/om/faculty/pinedo/scheduling/hurink/sl8.pdf`
[Dauzère-Pérès & Lasserre 93] S Dauzere-Peres, JB Lasserre, A modified shifting bottleneck procedure for job-shop scheduling, The international journal of production research 31 (4), 923-932, 1993
[Balas et al 1995] Egon Balas, Jan Karel Lenstra, Alkis Vazacopoulos. The One-Machine Problem with Delayed Precedence Constraints and its Use in Job Shop Scheduling, Management Science, 41(1), 94-109, 1995.
[Binato et al 2001] Binato, Silvio & Hery, W & Loewenstern, David & Resende, Mauricio. A greedy randomized adaptive search procedure for job shop scheduling, AT&T Labs Research Technical Report: 00.6.2, `https://www.researchgate.net/publication/228792021`
[Nowicki, Smutnicki 96] Nowicki, E. and C. Smutnicki, "A fast tabu search algorithm for the job-shop problem" Management Science, 42(6), 797–813 (1996).

[van Laarhoven et al 1992] Peter J. M. van Laarhoven, Emile H. L. Aarts, Jan Karel Lenstra: Job Shop Scheduling by Simulated Annealing. Oper. Res. 40(1): 113-125 (1992)
[Balas et Vazacopoulos 1998] Egon Balas, Alkis Vazacopoulos, Guided Local Search with Shifting Bottleneck for Job Shop Scheduling, Management Science 44(2), 262-275, 1998.
[Peng et al 2015] Peng, B., Lü, Z., & Cheng, T. C. E. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. Computers & Operations Research, 53, 154-164.
[Shen et al 2018] Liji Shen, Stéphane Dauzère-Pérès, Janis Sebastian Neufeld, Solving the flexible job shop scheduling problem with sequence-dependent setup times. Eur. J. Oper. Res. 265(2): 503-516 (2018)
[Dauzère-Pérès et Paulli 1997] Stéphane Dauzère-Pérès, Jan Paulli, An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann. Oper. Res. 70: 281-306 (1997)
[A. et Feillet 2008] Christian Artigues, Dominique Feillet, A branch and bound method for the job-shop problem with sequence-dependent setup times. Ann. Oper. Res. 159(1): 135-159 (2008)
[Roux et al. 1998] Stéphane Dauzère-Pérès, W. Roux, Jean B. Lasserre, Multi-resource shop scheduling with resource flexibility. Eur. J. Oper. Res. 107(2): 289-305 (1998)
[Tamssaouet et al 2018] Karim Tamssaouet, Stéphane Dauzère-Pérès, Claude Yugma, Metaheuristics for the job-shop scheduling problem with machine availability constraints. Comput. Ind. Eng. 125: 1-8 (2018)
[Knopp et al 2017] Sebastian Knopp, Stéphane Dauzère-Pérès, Claude Yugma. A batch-oblivious approach for Complex Job-Shop scheduling problems. Eur. J. Oper. Res. 263(1): 50-61 (2017)

[Gonzalez et al 2015] Miguel A. González, Angelo Oddi, Riccardo Rasconi, Ramiro Varela: Scatter search with path relinking for the job shop with time lags and setup times. Comput. Oper. Res. 60: 37-54 (2015)
[Patterson 1984] Patterson J. H., A comparison of exact approaches for solving the multiple constrained resource project scheduling problem, Management Science, vol. 30, num. 7, p. 854–867, 1984
[Alvarez-Valdes and Tamarit, 1989] Alvarez-Valdéz R., Tamarit J. M., Heuristic algorithms for resource-constrained project scheduling : A review and an empirical analysis, Slowinski R., Weglarz J., Eds., Advances in project scheduling, p. 113–134, Elsevier, 1989.
[Kolisch, Sprecher and Drexl 1995] R Kolisch, A Sprecher, A Drexl, Characterization and generation of a general class of resource-constrainted project scheduling problems Management science 41 (10), 1693-1703, 1995.
[Kolisch and Sprecher 1997] Kolisch R., Sprecher A., PSPLIB – A project scheduling library, European Journal of Operational Research, vol. 96, num. 1, p. 205–216, 1997.
[Baptiste and Le Pape 2000] Baptiste P., Le Pape C., Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems, Constraints, vol. 5, num. 1–2, p. 119–139, 2000.
[Carlier and Néron 2000] Carlier J., NÉRON E., On Linear Lower Bounds for the Resource Constrained Project Scheduling Problem, European Journal of Operational Research, vol. 149, p. 314–324, 2003.
[Kolisch & Hartmann 2006] Rainer Kolisch, Sönke Hartmann, Experimental investigation of heuristics for resource-constrained project scheduling: An update. Eur. J. Oper. Res. 174(1): 23-37 (2006)

[A. & Rivreau 2008] Christian ArtiguesDavid Rivreau. Heuristics. in: Christian Artigues, Sophie Demassey, Emmanuel Néron, editors, Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications, Wiley, 2008.
[Bartusch et al. 1988] M. Bartusch, R. H. Möhring & F. J. Radermacher, Scheduling project networks with resource constraints and time windows. Annals of Operations Research volume 16, pages 199–240 (1988)
[Alvarez-Valdés and Tamarit 1993] Alvarez-Valdéz R., Tamarit J. M., The project scheduling polyhedron : dimension, facets and lifting theorems, European Journal of Operational Research, vol. 67, num. 2, p. 204–220, 1993.
[Balas 1985] Balas E., On the facial structure of scheduling polyhedra, Mathematical Programming Study, vol. 24, p. 179–218, 1985.
[A. et al 2003] C. Artigues, P. Michelon, and S. Reusser. Insertion techniques for static and dynamic resource constrained project scheduling. European Journal of Operational Research, 149(2) :249-267, 2003.
[A. et Briand 2009] Christian Artigues, Cyril Briand, The resource-constrained activity insertion problem with minimum and maximum time lags. J. Sched. 12(5): 447-460 (2009)
[Borreguero et al. 2015] Borreguero, T., Artigues, C., Garcıa Sanchez, A., Ortega Mier, M., Lopez, P. (2015). Multimode time-constrained scheduling problems with generalized temporal constraints and labor skills. 7th multidisciplinary international conference on scheduling: Theory and applications (MISTA) (pp. 809–813). Prague, Czech Republic.
[Borreguero et al. 2021] T. Borreguero, T. Portoleau, A. Garcia Sanchez, M. Ortega Mier, and P. Lopez. Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective. Technical report. `https://hal.laas.fr/hal-03344445/document`, 2021.

[Simonin et al., 2012] Gilles Simonin, Christian Artigues, Emmanuel Hebrard, Pierre Lopez, Scheduling Scientific Experiments on the Rosetta/Philae Mission. CP 2012: 23-37
[Simonin et al., 2015] Gilles Simonin, Christian Artigues, Emmanuel Hebrard, Pierre Lopez: Scheduling scientific experiments for comet exploration. Constraints An Int. J. 20(1): 77-99 (2015)
[Queyranne and Shulz 1994] Queyranne M., Schulz A., Polyhedral approaches to machine scheduling, Report num. 408/1994, Technischen Universität Berlin, 1994.
[Applegate and Cook 1991] Applegate D., Cook W., A computational study of job-shop scheduling, ORSA Journal on Computing, vol. 3, num. 2, p. 149–156, 1991.
[Dyer and Wolsey 1990] Dyer M. E., Wolsey L. A., Formulating the single machine sequencing problem with release dates as a mixed integer program, Discrete Applied Mathematics, vol. 26, p. 255–270, 1990.
[Demassey et al. 2005] Demassey S, Artigues C, Michelon P. Constraint propagation-based cutting planes : An application to the resource-constrained project scheduling problem. INFORMS Journal on Computing 17(1) :52–65, 2005.
[Lasserre and Queyranne 1992] J.-B. Lasserre and M. Queyranne. Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling. In E. Balas, G. Cornuéjols, and R. Kannan, editors, Integer Programming and Combinatorial Optimization, pages 136–149. Carnegie Mellon University, 1992. Proceedings of the 2nd International IPCO Conference
[Dauzère-Pérès and Lasserre 1995] S. Dauzeère-Pérès and J.-B. Lasserre. A new mixed-integer formulation of the flow-shop sequencing problem. Paper presented at the Second Workshop on Models and Algorithms for Planning and Scheduling Problems, Wernigerode, Germany, May 1995.

[Pinto and Grossmann 1995] Pinto, J. M. ; Grossmann, I. E. A. Continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research 34 (9), 3037–3051, 1995.
[Zapata el al 2008] J. C. Zapata, B. M. Hodge, and G. V. Reklaitis. The multimode resource constrained multiproject scheduling problem : Alternative formulations, AIChE Journal, 54(8) : 2101–2119, 2008.
[Koné et al 2011] O. Koné, C. Artigues, P. Lopez, and M. Mongeau. Event-based MILP models for resource-constrained project scheduling problems. Computers and Operations Research, 38(1) :3–13, 2011.
[Koné et al 2013] Christian Artigues, Peter Brucker, Sigrid Knust, Oumar Koné, Pierre Lopez, Marcel Mongeau: A note on "event-based MILP models for resource-constrained project scheduling problems". Comput. Oper. Res. 40(4): 1060-1063 (2013)
[Tesch 2020] Alexander Tesch, A polyhedral study of event-based models for the resource-constrained project scheduling problem. J. Sched. 23(2): 233-251 (2020)
[Nattaf, Kis, A., Lopez 2019] Margaux Nattaf, Markó Horváth, Tamás Kis, Christian Artigues, Pierre Lopez: Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem. Discret. Appl. Math. 258: 188-203 (2019)
[Pritsker et al. 1969] Pritsker A. A., Watters L. J., Wolfe P. M., Multi-project scheduling with limited resources : a zero-one programming approach, Management Science, vol. 16, p. 93–108, 1969.
[Christofides et al. 1987] Christofides N., Alvarez-Valdéz R., Tamarit J. M., Project scheduling with resource constraints : a branch and bound approach, European Journal of Operational Research, vol. 29, num. 3, p. 262–273, 1987.

[Möhring et al. 2003] Möhring R., Schulz A., Stork F., Uetz M., Solving project scheduling problems by minimum cut computations, Management Science, vol. 49, num. 3, p. 330–350, 2003.
Christian Artigues, On the strength of time-indexed formulations for the resource-constrained project scheduling problem. Oper. Res. Lett. 45(2): 154-159 (2017)
[Klein 2000] Klein R. Scheduling of resource-constrained projects. Kluwer Academic Publishers, Dordrecht. 2000.
[Bianco and Caramia 2013] Bianco L and Caramia M. A new formulation for the project scheduling problem under limited resources. Flexible Services and Manufacturing Journal 25 :6–24, 2013.
[Hardin et al. 2008] Hardin JR, Nemhauser GL and Savelsbergh MW. Strong valid inequalities for the resource-constrained scheduling problem with uniform resource requirements. Discrete Optimization 5(1) :19–35, 2008.
[de Sousa and Wolsey 1997] de Souza CC, Wolsey LA. Scheduling projects with labour constraints. Relatório Técnico IC-P7-22. Instituto de Computação, Universidade Estadual de Campinas, 1997.
[Cavalcante et al. 2001] Cavalcante CCB, de Souza CC , Savelsbergh MWP, Wang Y, Wolsey LA. Scheduling projects with labor constraints. Discrete Applied Mathematics 112(1–3) :27–52, 2001.
[Baptiste and Demassey 2004] Baptiste P, Demassey S. Tight LP bounds for resource constrained project scheduling. OR Spectrum 26 (2), 251–262, 2004.
[Mingozzi et al. 1998] Mingozzi A, Maniezzo V, Ricciardelli S, Bianco L. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Manage Science 44 :714–729, 1998.

[Brucker and Knust 2000] Brucker P., Knust S. A linear programming and constraint propagation-based lower bound for the RCPSP, European Journal of Operational Research, vol. 127, p. 355–362, 2000.
[Demassey et al. 2004] S. Demassey, C. Artigues, P. Baptiste, and P. Michelon. Lagrangean relaxation-based lower bounds for the RCPSP. In 8th International Workshop on Project Management and Scheduling, pages 76-79, Nancy, France, 2004.
[Moukrim et al 2013] A Moukrim, A Quilliot, H Toussaint : Branch and Price for Preemptive Resource Constrained Project Scheduling Problem Based on Interval Orders in Precedence Graphs. FedCSIS 2013 : 321-328, 2013.
[Polo et al 2020] Oliver Polo-Mejía, Christian Artigues, Pierre Lopez, Virginie Basini, Mixed-integer/linear and constraint programming approaches for activity scheduling in a nuclear research facility. Int. J. Prod. Res. 58(23): 7149-7166 (2020)
[Demeulemeester & Herroelen (1997)] E. Demeulemeester and W. Herroelen, New benchmark results for the resource-constrained project. scheduling problem, Management Science 43 (1997)
[Tamura, Tanjo & Banbara 2012] Tomoya Tanjo, Naoyuki Tamura, Mutsunori Banbara, Azucar: A SAT-Based CSP Solver Using Compact Order Encoding - (Tool Presentation). SAT 2012: 456-462
[Schutt et al. 2009] Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, Mark Wallace, Why Cumulative Decomposition Is Not as Bad as It Sounds. CP 2009: 746-761
[Schutt et al. 2011] Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, Mark G. Wallace, Explaining the cumulative propagator. Constraints An Int. J. 16(3): 250-282 (2011)
[Schutt et al. 2013] Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, Explaining Time-Table-Edge-Finding Propagation for the Cumulative Resource Constraint. CPAIOR 2013: 234-250

[Katsirelos & Bacchus 2005]George Katsirelos, Fahiem Bacchus, Generalized NoGoods in CSPs. AAAI 2005: 390-396
[Gay et al 2015] Steven Gay, Renaud Hartert, Christophe Lecoutre, Pierre Schaus: Conflict Ordering Search for Scheduling Problems. CP 2015: 140-148
[Vilím et al 2015] Petr Vilím, Philippe Laborie, Paul Shaw: Failure-Directed Search for Constraint-Based Scheduling. CPAIOR 2015: 437-453
[Siala et al 2015] Mohamed Siala, Christian Artigues, Emmanuel Hebrard: Two Clause Learning Approaches for Disjunctive Scheduling. CP 2015: 393-402
[Laborie 2005] Philippe Laborie: Complete MCS-Based Search: Application to Resource Constrained Project Scheduling. IJCAI 2005: 181-186