Méthodes exactes pour l'ordonnancement

Christian Artigues

LAAS-CNRS
artigues@laas.fr

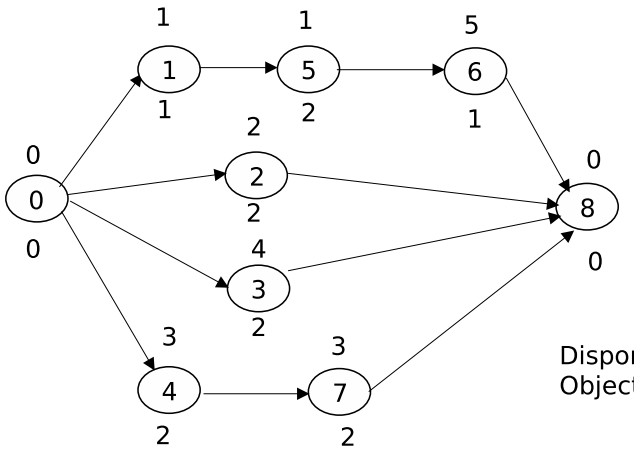
Ecole Jeunes Chercheur.e.s en Ordonnancement du GDR Recherche Opérationnelle

20 novembre 2017

L'ordonnancement de tâches

Définition informelle

Ordonnancer c'est organiser dans le temps l'exécution d'un ensemble de tâches soumises à des contraintes de temps et de ressources afin d'optimiser un objectif.



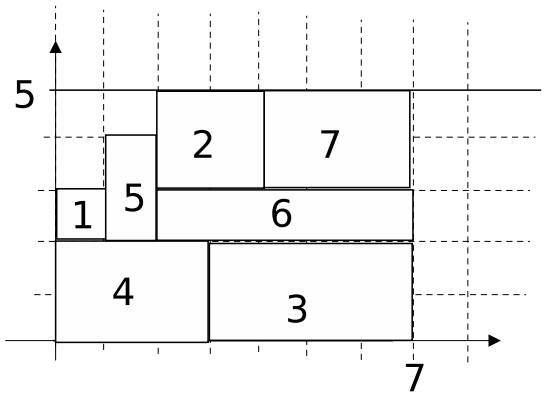
duree(i)

i

quantité requise(i)

Disponibilité de la ressource = 5
Objectif = minimiser la durée totale

Solution optimale



Types de problèmes d'ordonnancement

Ordonnancement statique/dynamique

Lorsque les données d'un problème d'ordonnancement sont connues à l'avance on parle de problème statique. Si les données apparaissent au cours du temps on parle de problème dynamique.

Ordonnancement déterministe/sous incertitudes

Si à partir du moment où la donnée est connue, elle l'est avec certitude on parle de problème déterministe, sinon on parle de problème stochastique et on peut modéliser les données par des variables aléatoires ou tout autre représentation de l'incertain (intervalles, ...).

Ordonnancement acyclique/cyclique

Lorsque les tâches à ordonnancer doivent être répétées indéfiniment on parle de problème cyclique. Sinon, on parle de problème acyclique.

Ordonnancement mono-objectif/multi-objectif

Lorsqu'il y a plusieurs fonctions objectifs à optimiser, on peut se ramener à une fonction mono-objectif ou chercher un ensemble de solutions dominanites au sens de Pareto [voir cours Vincent T'kindt].

Ordonnancement acyclique, statique, déterministe, mono-objectif

Un problème d'ordonnancement acyclique, statique et déterministe est un problème d'optimisation combinatoire.

Optimisation combinatoire

Etant donnés :

- Un ensemble discret X,
- une fonction $f : X \rightarrow \mathbb{R}$,

un problème d'optimisation combinatoire consiste à déterminer :

$$\min\{f(x) : x \in X\}$$

Comment résoudre un problème d'ordonnancement

Schéma classique de résolution des problèmes d'optimisation combinatoire

Détermination de la complexité [Voir Cours Ph. Chrétienne]

- Trouver un algorithme polynomial en fonction de la taille de l'entrée, ou bien
- Montrer que le problème est NP-difficile

Résolution d'un problème NP-difficile

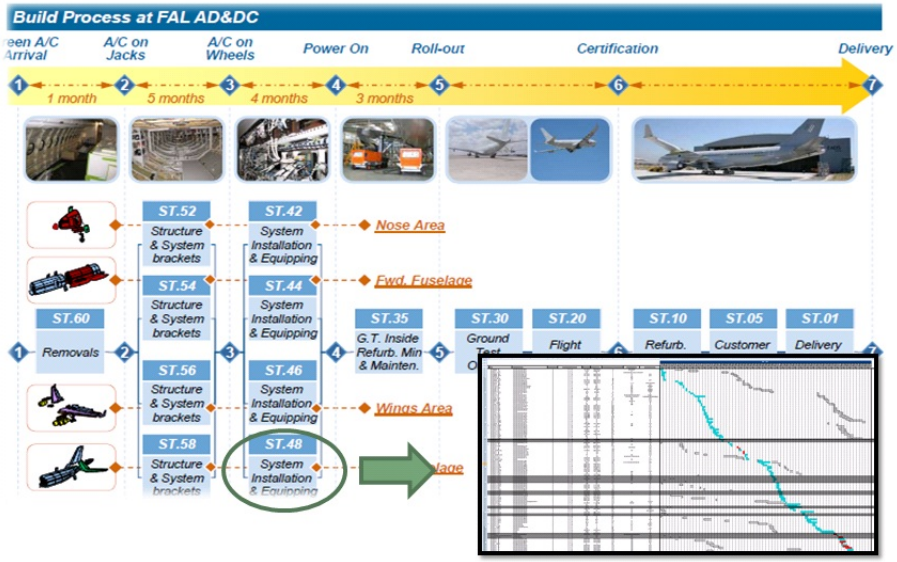
- Si NP-difficile au sens faible, il existe un algorithme pseudo-polynomial (polynomial pour un codage unaire de l'entrée)
- Si NP-difficile au sens fort ou si algorithme pseudopolynomial non utilisable en pratique :
  - Méthodes exactes (programmation linéaire en nombres entiers, programmation dynamique, procédure de séparation et d'évaluation, PPC [voir cours Pierre Lopez], SAT) ou approchées (heuristiques, metaheuristiques) [Voir Cours Stéphane Dauzère-Pérès].

Classification des problèmes d'ordonnancement

La notation $\alpha|\beta|\gamma$

- α décrit le type de ressources et leur organisation
  - $\alpha = 1$: problème à une machine [voir cours Ph. Chrétienne]
  - $\alpha = P$: problèmes à machines parallèles [voir cours Ph. Chrétienne]
  - $\alpha = F$: problème d'atelier (flow-shop) [voir cours Ph. Chrétienne]
  - $\alpha = J$: problème d'atelier (job-shop) [voir cours Ph. Chrétienne]
  - $\alpha = PS$: problème à ressources cumulatives (ou discrètes) : RCPSP
- β précise les caractéristiques des tâches
  - $pmtn \in \beta$: possibilité de préemption (interruption et reprise des tâches)
  - $r_i \in \beta$: problème avec dates de disponibilités
  - $\tilde{d}_i \in \beta$: problème avec deadlines
  - $p_i = 1 \in \beta$: problème à durées unitaires
  - $prec \in \beta$: problème à contraintes de précédence simples ($l_{ij} = p_i$)
  - $temp \in \beta$: problème à contraintes de précédence généralisées
- γ décrit la fonction objectif

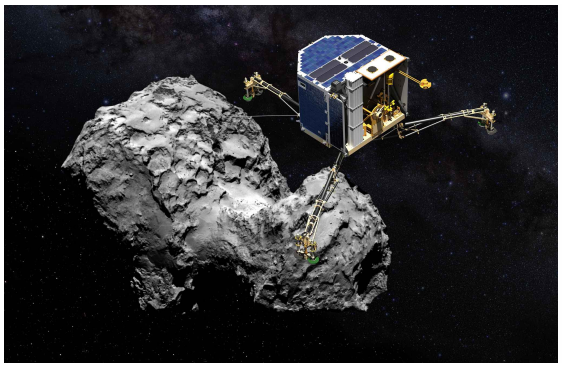
Exemple d'application 1/3
Chaine d'assemblage de l'Airbus A330 MRT

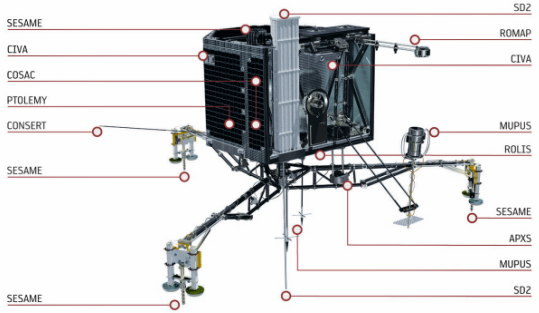


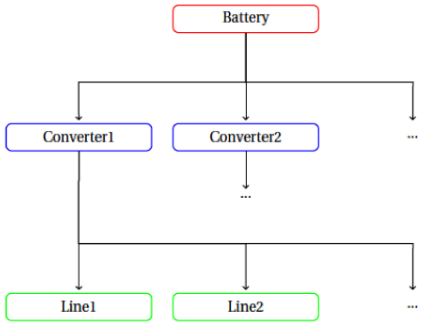
- RCPSP à date de fin imposée (*takt time*), objectif minimiser le nombre maximal de ressources utilisées
- Environ 600 tâches
- Ressources : opérateurs
- Problème multi-modes

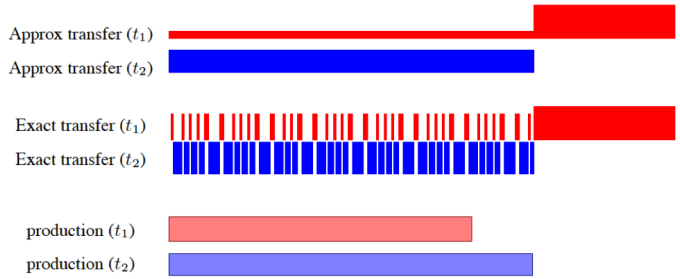
[Borreguero et al., 2015]

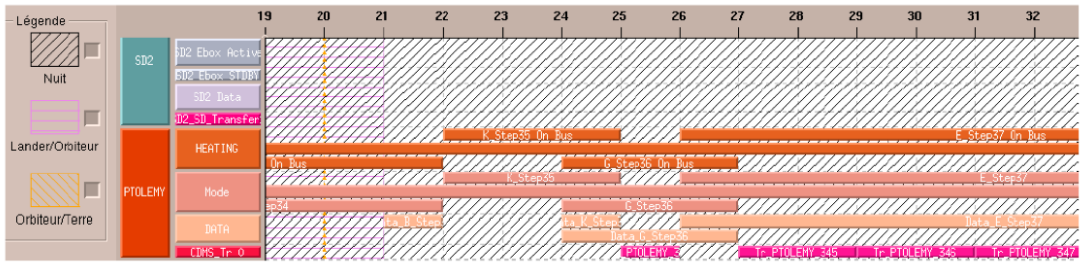
Exemple d'application 2/3 : ordonnancement des expériences de Philae sur la comète «Tchouri»











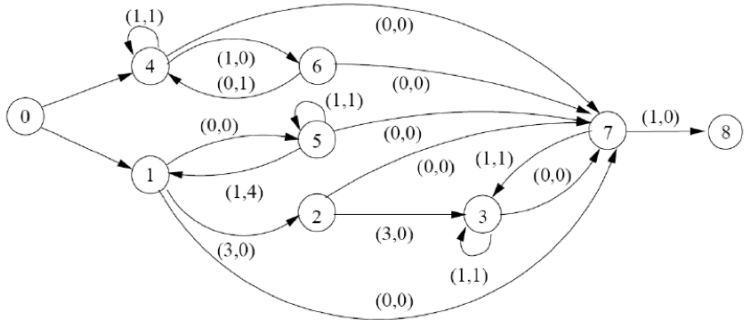
- RCPSP avec contraintes de transferts de données
- Hiérarchie de ressources cumulatives à trois niveaux
- 19 expériences, 752 tâches, 168 événements, 926 précédences,

[Simonin et al., 2012 2015]

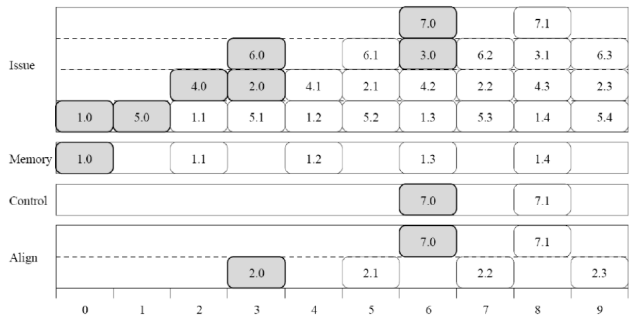
Exemple d'application 3/3 : ordonnancement d'instructions sur processeur VLIW ST200

```
int
prod(int n,
     short a[],
     short b)
{
   int s=0, i;
   for (i=0; i<n; i++) {
      s += a[i]*b;
   }
   return s;
}
```

```
L?__0_8:
LDH_1    g131 = 0, G127
MULL_2   g132 = G126, g131
ADD_3    G129 = G129, g132
ADD_4    G128 = G128, 1
ADD_5    G127 = G127, 2
CMPNE_6 b135 = G118, G128
BRF_7    b135, L?__0_8
```



Resource	ISSUE	MEM	CTL	ODD	EVEN	LANE0
Availability	4	1	1	2	2	2
ALL	4	0	0	0	0	0
ALU	1	0	0	0	0	0
ALUX	2	0	0	1	1	0
CTL	1	0	1	1	1	0
ODD	1	0	0	1	0	0
ODDX	2	0	0	1	1	0
MEM	1	1	0	0	0	0
MEMX	2	1	0	1	1	0
PSW	1	1	1	1	1	0
EVEN	1	0	0	0	1	0



- RCPSP cyclique (ordonnancement d'une boucle de grande taille)
- Durées unitaires, recherche d'une période optimale
- 214 instructions, 1063 précédences [Ayala et al. 2013]

Algorithmes (exacts) polynomiaux

- Algorithmes spécifiques [voir cours Philippe Chrétienne]
- Algorithmes d'affectation
- Algorithmes de flots
- Programmation linéaire
- Algorithmes de programmation dynamique polynomiaux

Plus grand retard avec préemption, dates de lancement et précédences $1|prec, pmtn, r_i|L_{\max}$ (1/2)

Définition

Ordonnancer un ensemble de tâches définies par des durées p_i, des dates de lancement r_i, des dates de livraison d_i, des contraintes de précédence simples (graphe $G(V, E)$), sur une machine en minimisant le plus grand retard $L_{\max} = \max_{i \in T}(C_i - d_i)$ avec préemption autorisée.

Modification des dates de lancement et de livraison $O(n + |E|)$

Les dates de lancement et de livraison peuvent être mises à jour en utilisant les contraintes de précédence comme suit :

$$r'_i = \max\left(r_i, \max_{i \in \sigma^-(i)}(r'_i + p_i)\right) \quad d'_i = \min\left(d_i, \min_{i \in \sigma(i)}(d'_j - p_j)\right)$$

où $\sigma^-(i)$ ($\sigma(i)$) désignent les prédécesseurs (successeurs) de i.

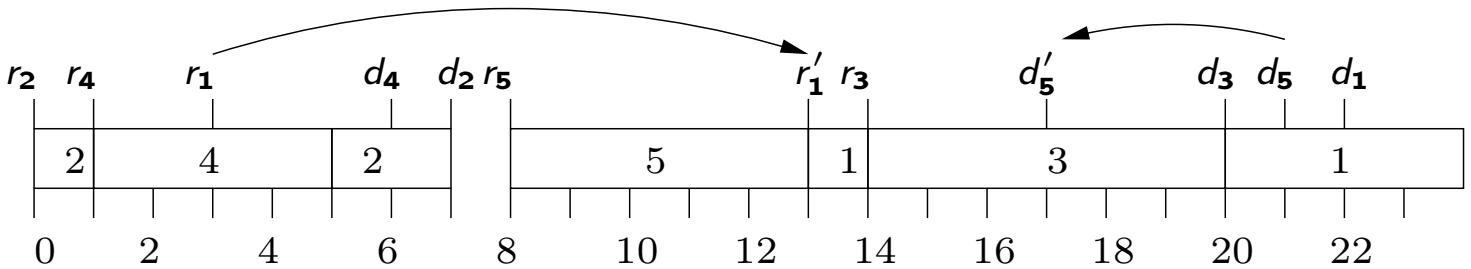
Plus grand retard avec préemption dates de lancement et précédences $1|prec, pmtn, r_i|L_{\max}$ $1|prec, pmtn, r_i|L_{\max}$ (2/2)

Résolution Algorithme de Jackson préemptif $O(n^2)$

A chaque instant t défini par une date de lancement modifiée ou la fin d'une tâche, ordonnancer la tâche de plus petite date de lancement modifiée en interrompant si nécessaire la tâche en cours à t.

preuve d'optimalité par argument d'échange.

i	p_i	r_i	d_i	$\sigma(i)$
1	5	3	22	–
2	3	0	7	–
3	6	14	20	–
4	4	1	6	–
5	5	8	21	1



Fonction régulière de type somme avec durées unitaires et dates de lancement $1|r_i, p_i = 1|\sum f_i$

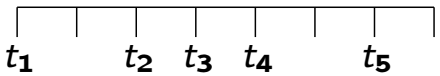
Définition

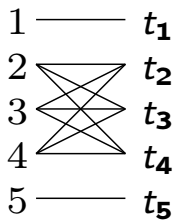
Ordonnancer sur une machine un ensemble de tâches définies par des durées $p_i = 1$, des dates de lancement r_i, en minimisant une fonction régulière de type somme $\sum_{i \in T} f_i$.

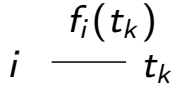
Précalcul des dates d'ordonnancement

Il existe un ordonnancement optimal tel que chaque tâche démarre à une des n dates définies récursivement par $t_1 = r_1$ et $t_i = \max(r_i, t_{i-1} + 1)$ (en supposant $r_1 \leq \ldots \leq r_n$).

i	r_i
1	0
2	3
3	3
4	3
5	6







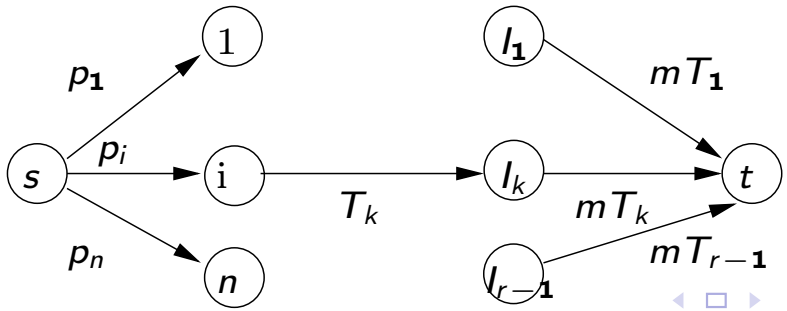
Résolution d'un problème d'affectation ($O(n^3)$)

Problème de solution réalisable avec dates de lancement, dates d'échéance et préemption $P|pmtn, r_i, \tilde{d}_i|-$

Définition

Ordonnancer sur m machines parallèles, n tâches de durées p_i, de dates de lancement r_i et dates d'échéances $\tilde{d}_i$, avec préemption autorisée.

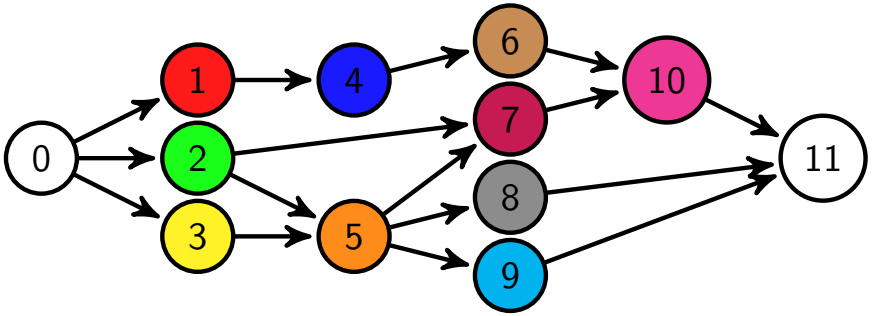
Le problème peut être transformé en problème de flot maximal. Soit $t_1 < t_2 < \ldots < t_r$ la séquence des valeurs différentes des r_i et des d_i. On considère les intervalles $I_k = [t_k, t_{k+1}]$ de longueur $T_k = t_{k+1} - t_k$. On crée un sommet par tâche et par intervalle ainsi qu'un sommet source s et un sommet puit t. On définit un arc de capacité T_k entre une tâche i et un intervalle tel que $r_i \leq t_k$ et $t_{k+1} \leq d_i$. On définit un arc de capacité p_i entre s et i et un arc de capacité mT_k entre I_k et t. Il existe une solution au problème $P|pmtn, r_i, \tilde{d}_i|-$ si et seulement s'il existe un flot maximal de valeur $\sum_{i=1}^{n} p_i$.



Programmation dynamique

- Diviser pour régner
- Principe d'optimalité : une sous-solution doit aussi être optimale pour le sous-problème.
- Etapes :
  - Découper le problèmes en étapes de prise de décision, basées sur la structure du problème
  - Trouver une relation de récurrence qui exprime la solution optimale d'un problème avec k étapes à décider en fonction des problèmes avec $k-1$ étapes à décider : formulation arrière
  - Trouver une relation de récurrence qui exprime la solution optimale d'un problème avec k étapes à décider en fonction des problèmes avec $k+1$ étapes à décider : formulation avant
  - Chercher une approche de résolution des sous-problèmes de manière ascendante pour éviter redondance.
- Algorithmes polynomiaux, pseudo-polynomiaux ou exponentiels

Ordonnancement sous contraintes de précédence



$p_1 = 3, p_2 = 5, p_3 = 1, p_4 = 3, p_5 = 2, p_6 = 4, p_7 = 5, p_8 = 6, p_9 = 4, p_{10} = 4,\ T = 24$

- $\mathcal{S}_T^{\emptyset}$: Ordos respectant les contraintes de précédence et la borne supérieure T.

Formulation primale

min $S_{n+1} - S_0$ s.c.

$$S \in \mathcal{S}_T^{\emptyset} = \left\{ \begin{array}{l} S_{n+1} - S_0 \leq T \\ S_j - S_i \geq +p_i, (i,j) \in E \\ S_0 = 0 \end{array} \right\}$$

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

Formulation duale

max $\sum_{(i,j)\in E} p_j \phi_{i,j} - T\phi_{n+1,0}$ s.c.

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

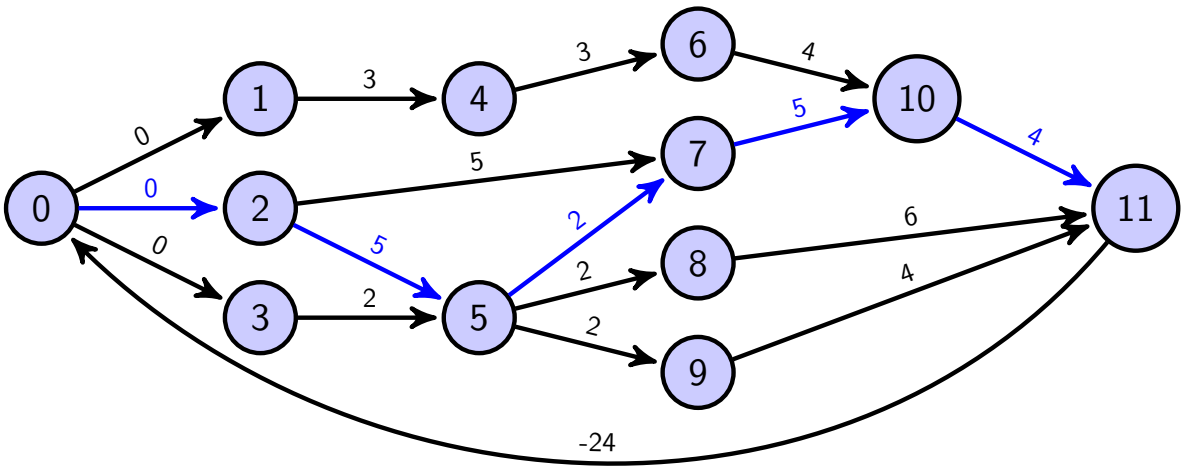
Problème de *plus long chemin* entre 0 et $n+1$.

- Graphe potentiels-tâches $G(A, E^+, V)$
  avec $E^+ = E \cup (n+1, 0)$ et $V_{i,j} = p_i$, $(i,j) \in E$, $V_{n+1,0} = -T$.
- Formulation récurrente $S_0^0 = 0$ et $S_j^k = \max(S_j^{k-1}, \max_{i \in A | (i,j) \in E} S_i^{k-1} + pi)$, $j \in A$, $k \geq 1$

  Se résout par programmation linéaire ou bien par l'algorithme de Bellman en $O(|E|)$.

Ordonnancements au plus tôt et au plus tard

- $L^*(i,j)$: longueur du plus long chemin entre i et j dans $G(A, E^*, V)$.
- ES : élément minimum de $\mathcal{S}_T^{\emptyset}$ tel que $ES_i = L^*(0,i)$
- LS : élément maximum de $\mathcal{S}_T^{\emptyset}$ tel que $LS_i = ES_{n+1} - L^*(i,n+1)$
- Formulation récurrente $LS_{n+1}^0 = T$ et
  $LS_j^k = \min(LS_j^{k-1}, \min_{i \in A | (j,i) \in E} LS_i^{k-1} - p_j)$, $j \in A$, $k \geq 1$
- On a $\mathcal{S}_T \subseteq \mathcal{S}_T^{\emptyset} \implies S_i \in [ES_i, LS_i], i \in A$, pour tout $S \in \mathcal{S}_T$

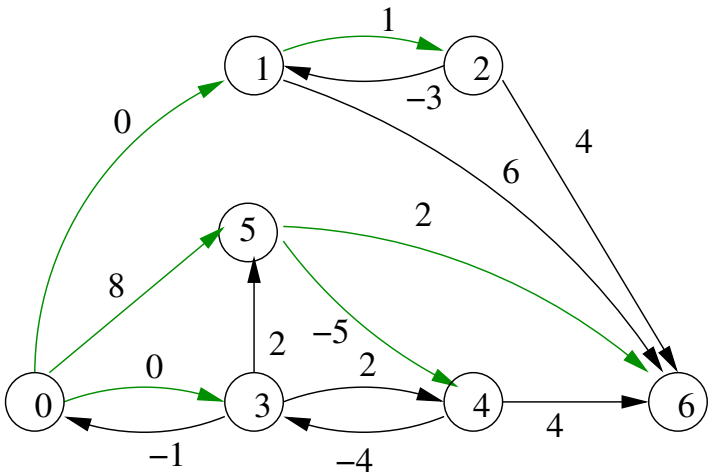


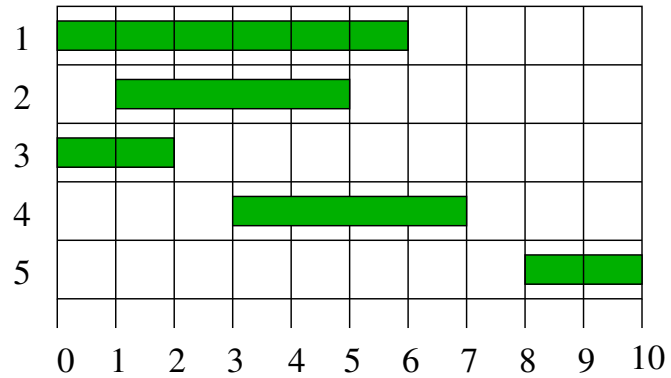
$T = 24$

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]

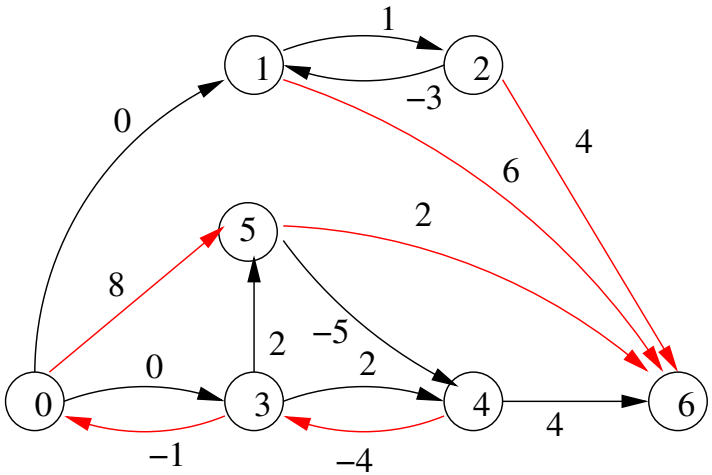
Programmation dynamique avant et arrère

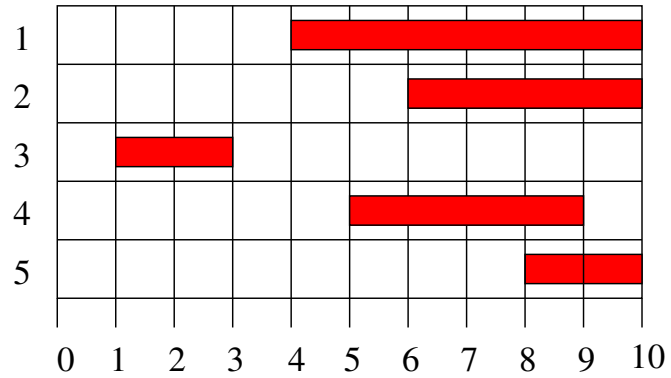
Ordonnancement au plus tôt et plus longs chemins correspondants





Ordonnancement au plus tard et plus longs chemins correspondants

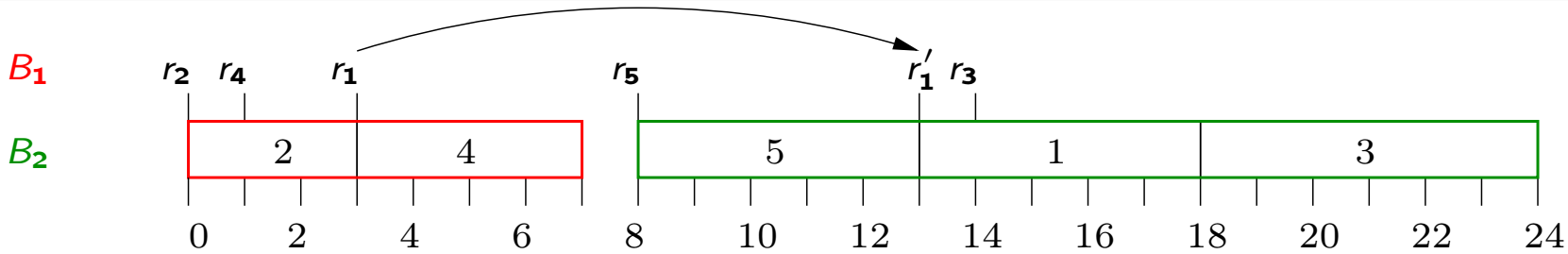




Fonction régulière de type maximum avec préemption et précédences $1|prec, pmtn, r_i|f_{\max}$ (1/3)

Blocs de tâches

Soit une solution non prémptive obtenue, en ordonnançant les tâches dans l'ordre croissant des dates de lancement modifiées. La solution obtenue est une succession de blocs de tâches consécutives $B_1, ..., B_b$ séparés par des périodes d'inactivité de la machine.



Théorème (décomposition du problème)

Il existe un ordonnancement préemptif optimal respectant l'ordre des blocs, les tâches étant exécutées sans temps d'inactivité à l'intérieur de chaque bloc. Si $f^*_{\max}(B)$ est la valeur optimale du problème réduit aux tâches de B alors la valeur optimale du problème est $f^*_{\max} = \max_{j=1,...,b} f^*_{max}(B_j)$.

Fonction régulière de type maximum avec préemption et précédences $1|prec, pmtn, r_i|f_{\max}$ (2/3)

Définition récurrente de la solution optimale d'un bloc

Soient :

- $S(B)$ date de début du bloc B.
- $p(B)$ durée totale du bloc B
- $C(B) = S(B) + p(B)$ date de fin du bloc B

Il existe un ordonnancement optimal dans lequel la tâche k sans successeur dans B telle que

$$f_k(C(B)) = \min_{i \in B, \sigma(i) \cap B = \emptyset} f_i(C(B))$$

se termine à $C(B)$. On a donc

$$f^*_{\max}(B) = \max\left(f^*_{\max}(B \setminus \{k\}), f_k(C(B))\right)$$

Fonction régulière de type maximum avec préemption et précédences $1|prec, pmtn, r_i|f_{\max}$ (3/3)

Algorithme récursif

Décompose(T)

1. Déterminer $B_1, \ldots, B_b$
2. $f \leftarrow -\infty$
3. Pour chaque bloc B faire
4. Déterminer k telle que $f_k(C(B)) = \min_{i \in B, \sigma(i) \cap B = \emptyset} f_i(C(B))$
5. $f \leftarrow \max\{f, f_k(C(B), \text{Décompose}(B \setminus \{k\})\}$
6. Retourner f

Résolution par appel de Décompose(T). Complexité $O(n^2)$.

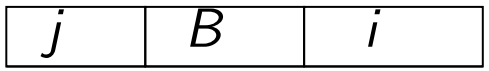
Remarque : cas $r_i = 0$ pour toute tâche i, la solution obtenue est non préemptive : optimal pour le $1|prec|f_{\max}$.

Somme pondérée du nombre de tâches en retard $1||\sum w_i U_i$ 1/3

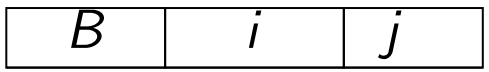
Le problème est NP-difficile (Karp, 1972). Le problème préemptif est donc également NP-difficile.

Caractérisation d'une solution optimale

Supposons $d_1 \leq \ldots \leq d_n$. Il existe un ordonnancement optimal donné par une séquence $i_1, i_2, \ldots, i_s, i_{s+1}, \ldots, i_n$ avec $i_1 < i_2 < \ldots < i_s$ où toutes les tâches $i_1, \ldots, i_s$ sont à l'heure et toutes les tâches $i_s + 1, \ldots, i_n$ sont en retard.



$d_i \leq d_j$



échange réalisable

On passe d'une espace de recherche à $n!$ éléments à un espace à 2^n éléments.

Somme pondérée du nombre de tâches en retard $1||\sum w_i U_i$ 2/3

Résolution par programmation dynamique.

Plongement dans une famille de sous-problèmes :
On définit $F_j(t)$ pour $j = 1, \ldots, n$ et $t = 0, \ldots, P = \sum_{i=1}^{n} p_i$, la valeur minimale du critère pour le problème d'ordonnancement des tâches $1, \ldots, j$ sachant que l'ordonnancement des tâches à l'heure se termine avant la date t. La valeur de la solution optimale est $F_n(\min(d_n, P))$.

Définition récurrente de $F_j(t)$

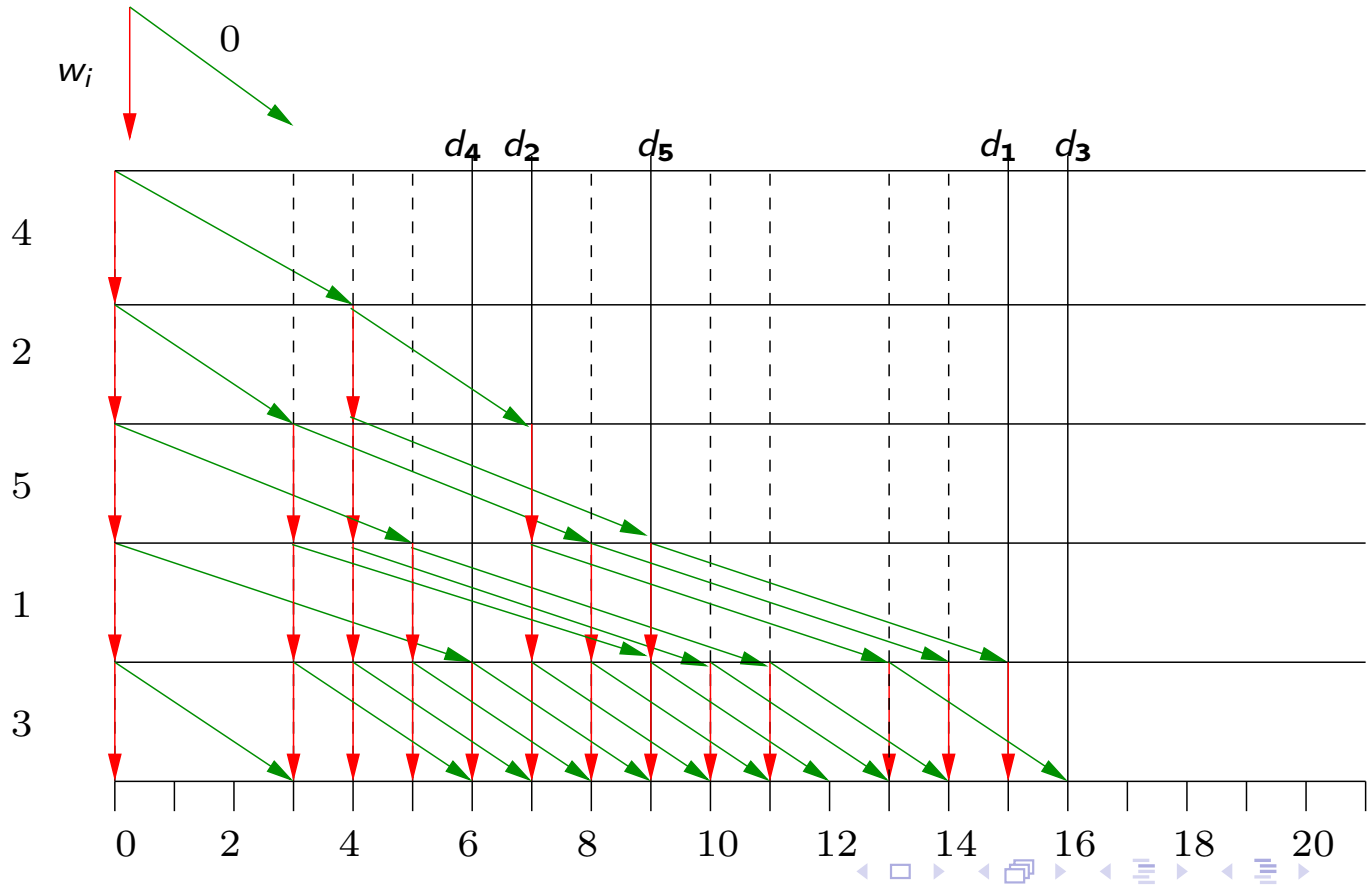
- Si $0 \leq t \leq d_i$ et $j \geq 1$ alors soit j est à l'heure à t et donc elle peut être placée à la fin de la séquence, soit j est en retard donc $F_j(t) = \min(F_{j-1}(t - p_j), F_{j-1}(t) + w_j)$.
- Si $t > d_j$ et $j \geq 1$ alors $F_j(t) = F_j(d_j)$ car toutes les tâches de $1, \ldots, j$ se terminant après d_j sont en retard.
- Si $t < 0$ ou $j = 0$ alors $F_j(t) = 0$

Somme pondérée du nombre de tâches en retard $1||\sum w_i U_i$ 2/3

Calcul ascendant des $F_j(t)$: $O(n \sum_{i=1}^{n} p_i)$ (algorithme pseudo polynomial).

Visualisation graphique de l'algorithme de programmation dynamique comme un plus court chemin

i	p_i	d_i
1	6	15
2	3	7
3	3	16
4	4	6
5	5	9



Algorithmes de séparation et d'évaluation (branch and bound)

- Egalement diviser pour régner : exemple branchement binaire $P = P_l \cup P_g$ et $P_l \cap P_g = \emptyset$.
- Séparation (branchement)
  - basée sur les dates $(S_i \geq 4 \wedge S_i \leq 3)$
  - basée sur les positions $(\wedge_{j \in Q} \sigma(j) = i)$
  - basée sur les précédences $(i \prec j \wedge j \prec i)$
  - autres [...]
- Evaluation
  - Algorithmes d'évaluation excès (BS pour un problème de min) : une heuristique
  - Algorithmes d'évaluation par défaut (BI pour un problème de min) : PL, relaxations spécifiques
  - Algorithmes d'inférence : enrichissement des contraintes (ex $i \not\prec j$) et/ou de réduction des domaines $(S_i = [ES_i, LS_i])$
  - Règles de dominance
- Élagage d'un nœud : $BI \geq BS$ ou infaisabilité (ex $S_= \emptyset$) ou dominance ou optimalité locale.

Plus grand retard avec dates de lancement $1|r_i|L_{\max}$ 1/3

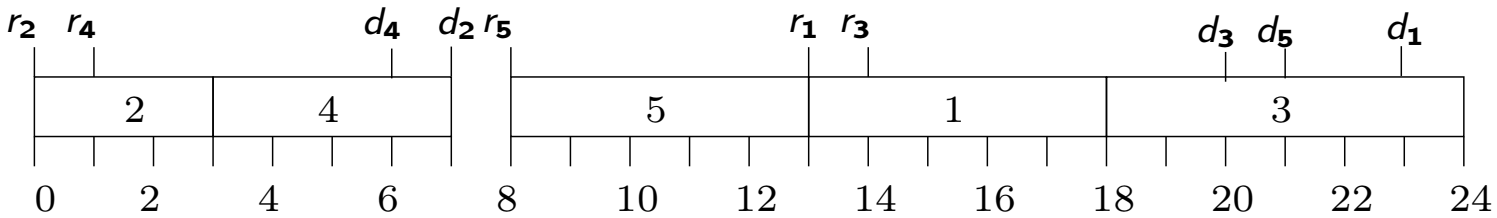
Le problème est NP-difficile (Lenstra et al. 1977).

Résolu efficacement par la procédure de séparation et d'évaluation de Carlier, 1982, basée sur l'analyse d'une solution approchée.

Algorithme de Jackson Non Préemptif (JNP)

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

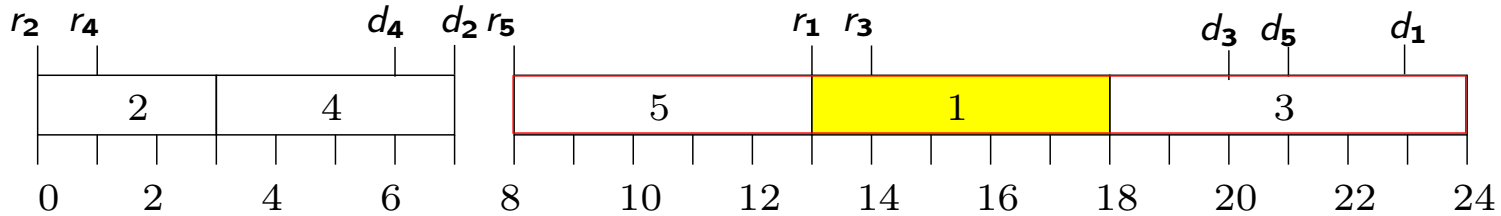


solution par l'algorithme de Jackson non préemptf $L_{\max} = 4$

Plus grand retard avec dates de lancement $1|r_i|L_{\max}$ 2/3

Analyse de la solution de Jackson non préemptive

Soit $i_1, \ldots, i_p$ le "chemin critique" tel que $L_{\max} = r_1 + \sum_{k=1,\ldots,p} p_{i_k} - d_{i_p}$. 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 $J = \{i_c + 1, \ldots, i_p\}$ Dans toute solution optimale i_c est exécuté, soit avant, soit après toutes les tâches de J.



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

Plus grand retard avec dates de lancement $1|r_i|L_{\max}$ 3/3

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)

Algorithme de séparation et d'évaluation pour le $J||C_{max}$

[voir Cours de Philippe Chrétienne]

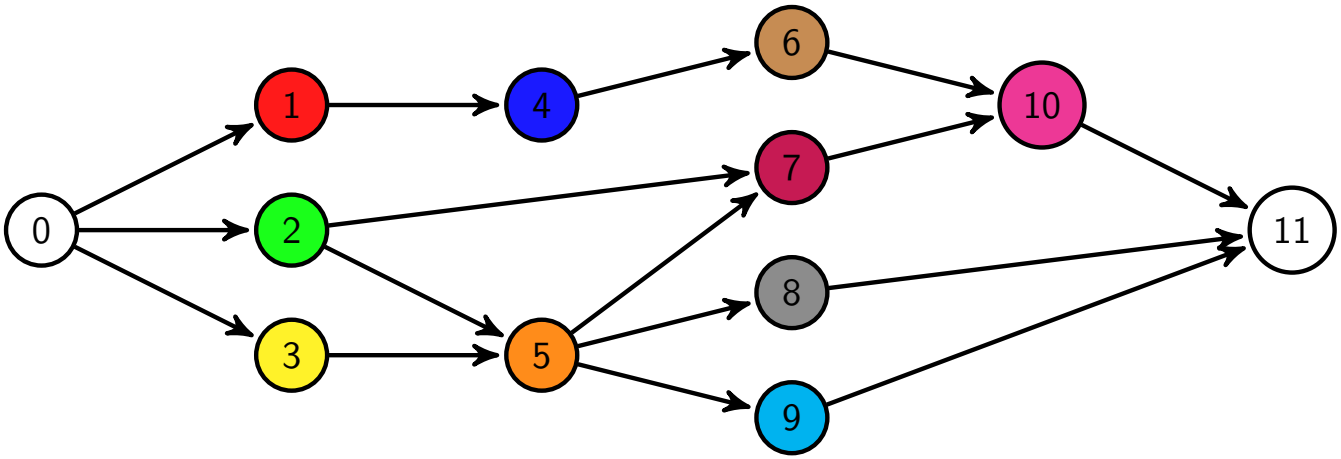
Le problème d'ordonnancement de projet à moyens limités

- 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].
  - 1 844 citations de PSPLIB (Google Scholar) 19/11/2017
  - 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} durée totale du projet

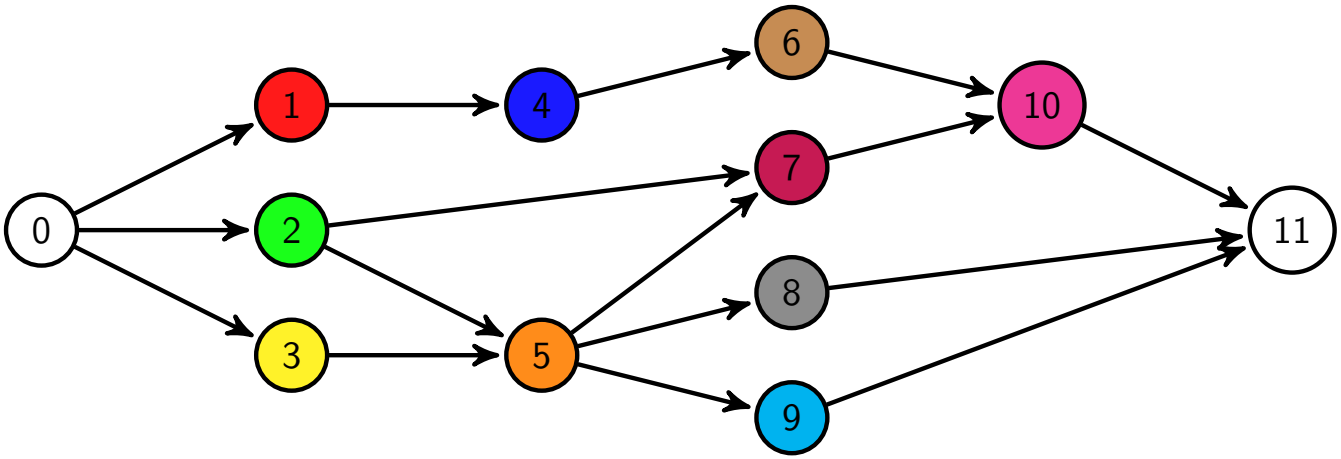
RCPSP (formulation conceptuelle)

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

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

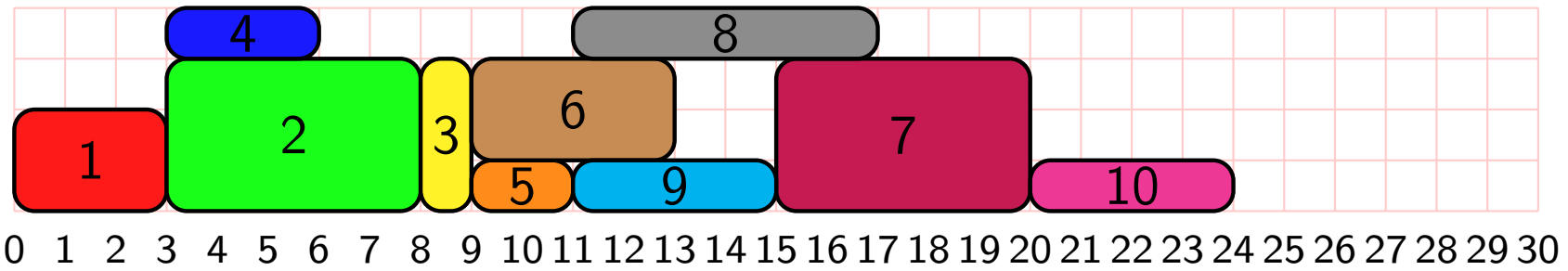
avec $A(t) = \{j \in A | t \in [S_j, S_j + p_j)\}, \forall t \in \mathcal{T}$

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

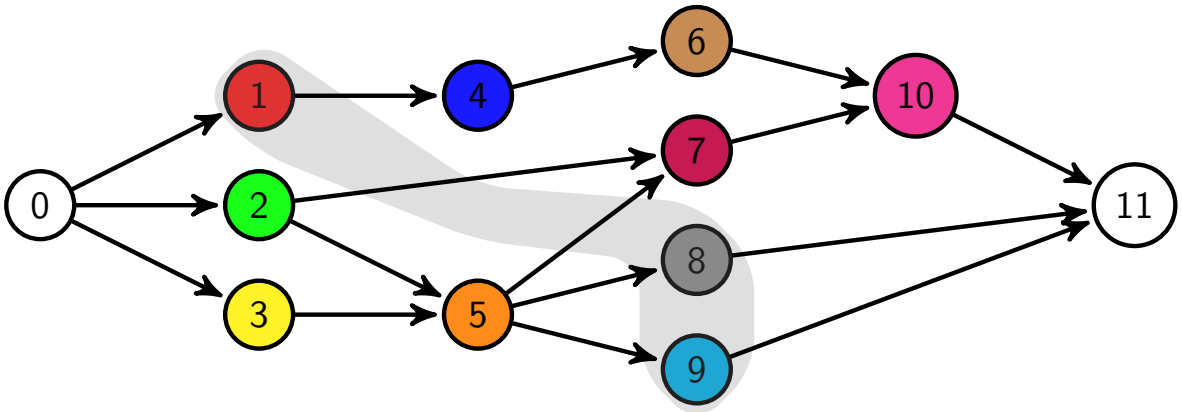


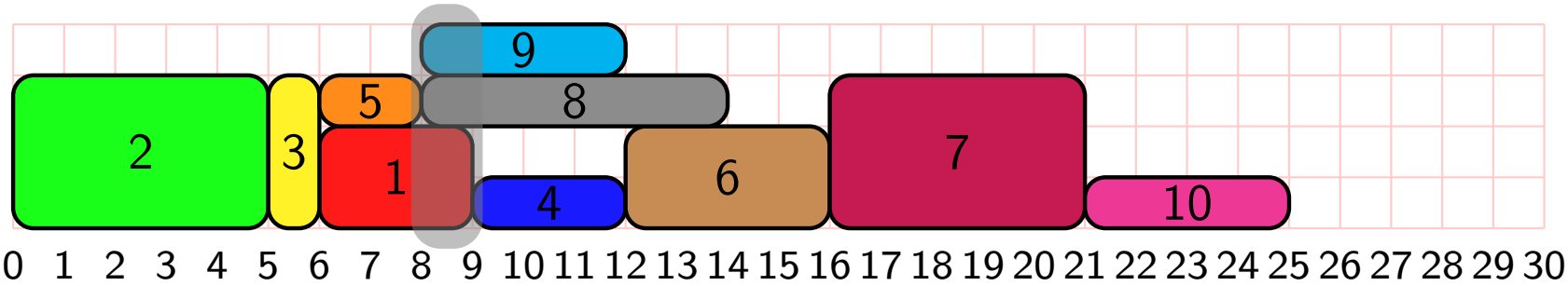
RCPSP : complexité, variantes and méthodes

- 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) [voir cours Pierre Lopez] ou hybridations SAT/CP

Antichaînes

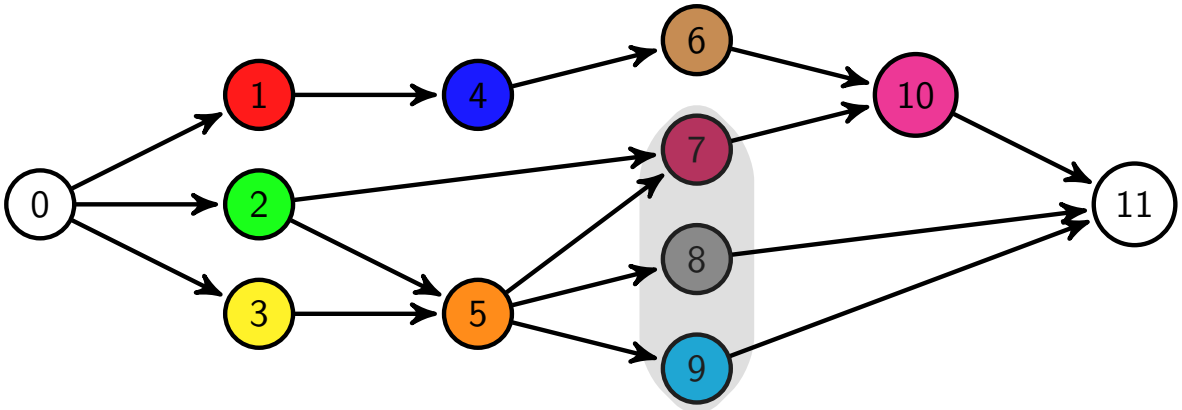
- Une antichaîne C de $G(A, E)$ est un ensemble de tâches non reliées deux à deux par un chemin.
- Pour toute antichaîne C, il existe une valeur T et un ordonnancement $S \in \mathcal{S}_T^{\emptyset}$ tels que $(\cap_{i \in C}[S_i, S_i + p_i) \neq \emptyset)$

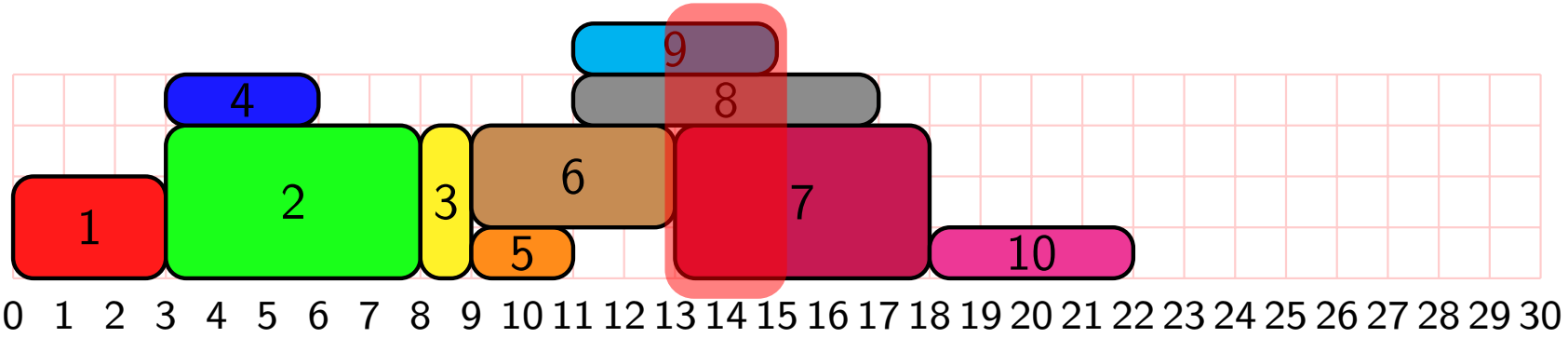




Antichaînes

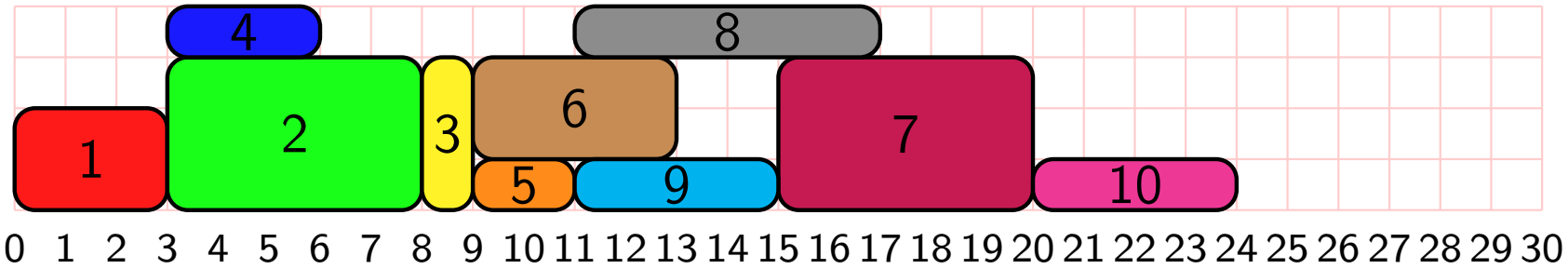
- Une antichaîne C de $G(A, E)$ est un ensemble de tâches non reliées deux à deux par un chemin.
- Pour toute antichaîne C, il existe une valeur T et un ordonnancement $S \in \mathcal{S}_T^{\emptyset}$ tels que $(\cap_{i \in C}[S_i, S_i + p_i) \neq \emptyset)$





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, $\exists i, j \in F$, $S_j \geq S_i + p_i$

Ordre strict et résolution des ensembles critiques minimaux

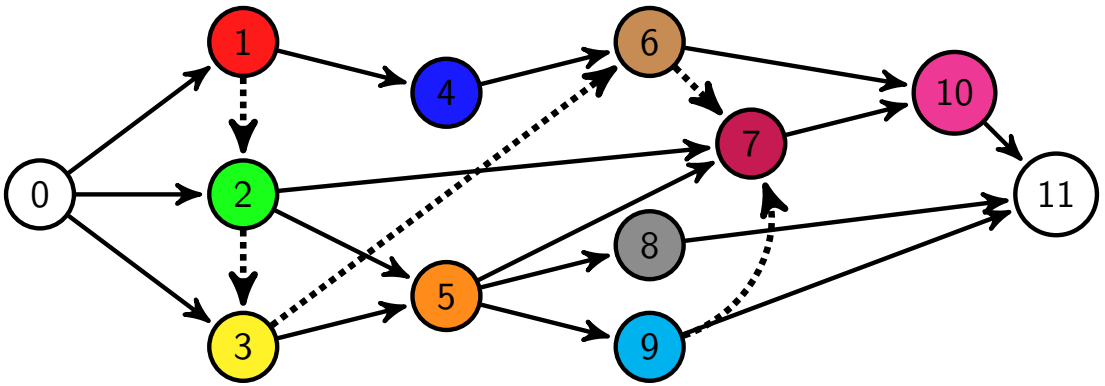
- X : un ensemble d'arcs «additionnels» : tel que $X \cap E = \emptyset$
- $G^X(A, E^+ \cup X, V)$: graphe potentiels-tâches incluant les arcs X.
- Soit $\prec^X$ la relation d'ordre strict sur A telle que :
  $i \prec^X j \Leftrightarrow$ «il existe dans G^X un chemin de longueur $\geq p_i$ de i vers 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).

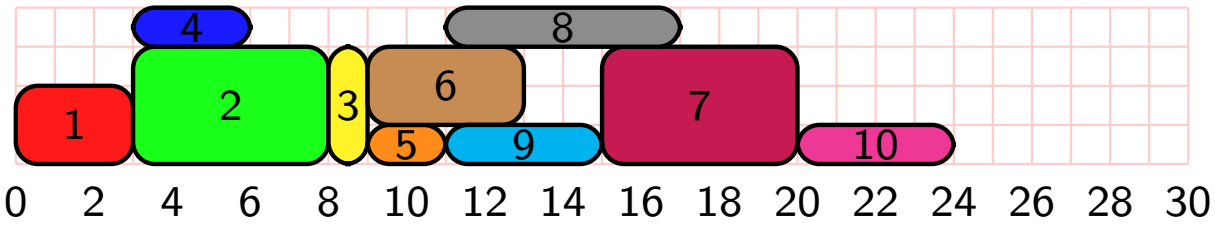
Représentation combinatoire de l'ensemble des solutions réalisables du RCPSP

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 $\mathcal{S}_T^X$.

Formulation du RCPSP et 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

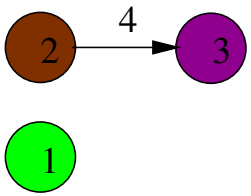
Algorithme BB-MFS

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: $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{X \cup \{(i,j)\}\}, \{X \cup \{(j,i)\}\}$

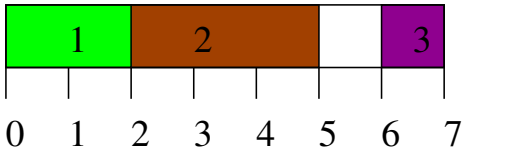
Evaluer(X,ES^X,LS^X,$UB-1$)

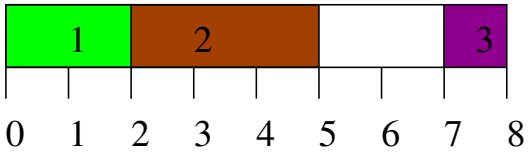
- Réduction des fenêtres *propagation de contraintes*
- Calcul de borne inférieure spécifiques ou par *programmation linéaire*
- Application de règles de dominance

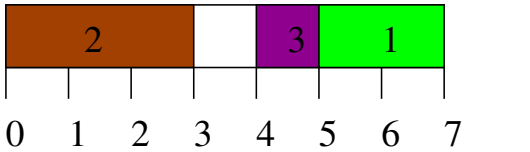
Exemple de règle de dominance : ordonnancements actifs et semi actifs

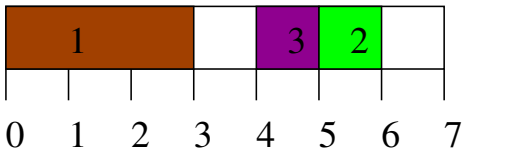


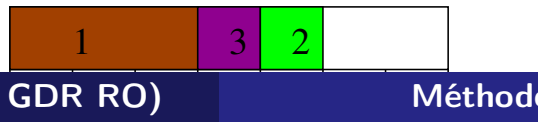
machine unique

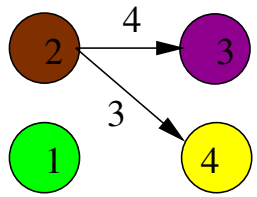




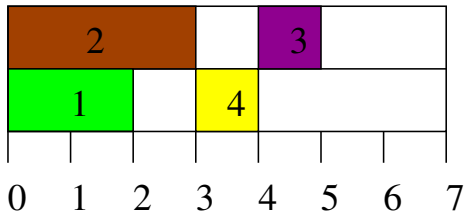


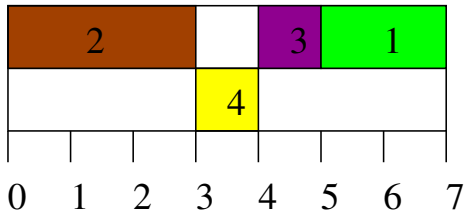


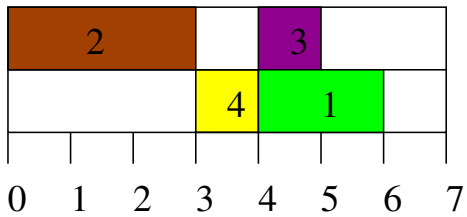


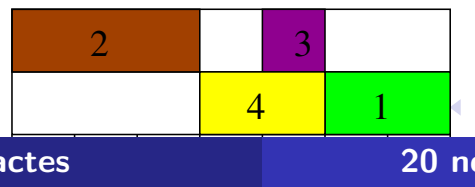


machines parallèles

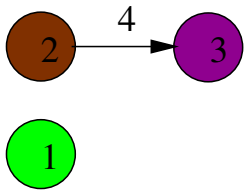




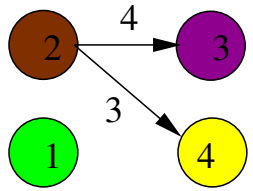




Exemple de règle de dominance : ordonnancements actifs et semi actifs

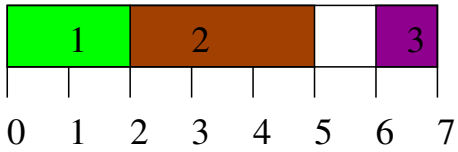


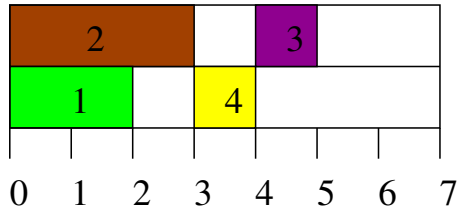
machine unique



machines parallèles

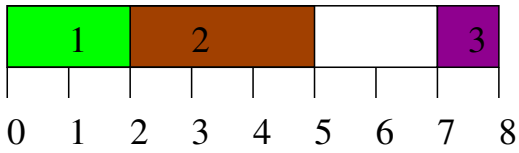
actif

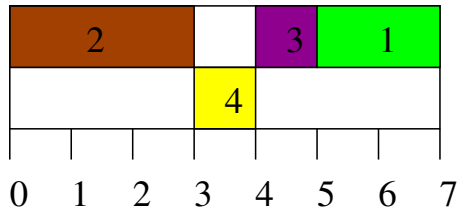




actif

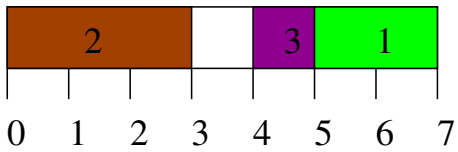
réalisable
non semi–actif

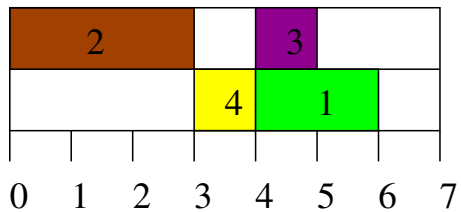




réalisable
non semi–actif

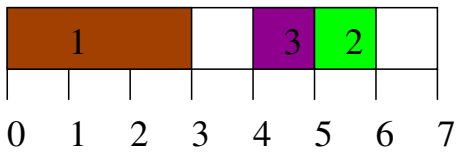
actif

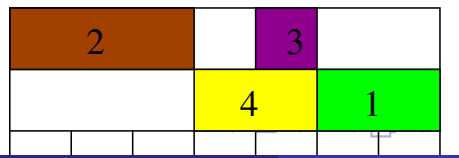




réalisable
non semi–actif
selon la définition

semi–actif
non actif



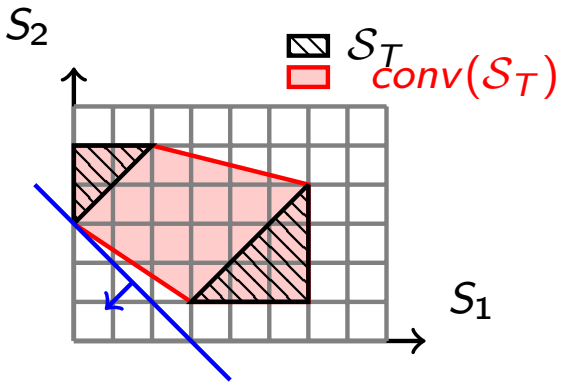


semi–actif
non actif

non réalisable!

Programmation linéaire en nombres entiers pour le RCPSP

- Formulation à variables entières additionnelles permettant d'exprimer les contraintes sous forme linéaire
- La projection de l'ensemble des solutions entières sur les variables S donne $\mathcal{S}_T$ ou un sous ensemble d'ordonnancements dominants
- La projection de l'ensemble des solutions continue sur les variables S est notée $\tilde{\mathcal{S}}_T$ définit la qualité de la relaxation avec idéalement $\tilde{\mathcal{S}}_T = conv(\mathcal{S}_T)$.



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}

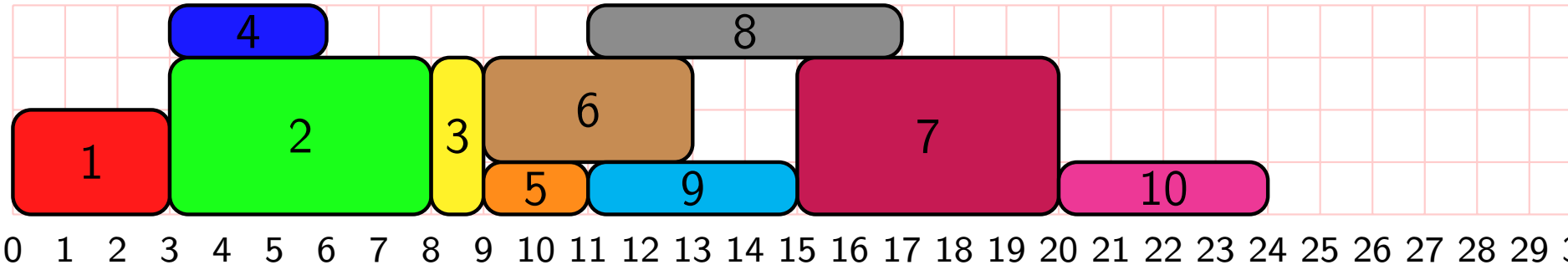
Variables de dates de début et d'ordre strict

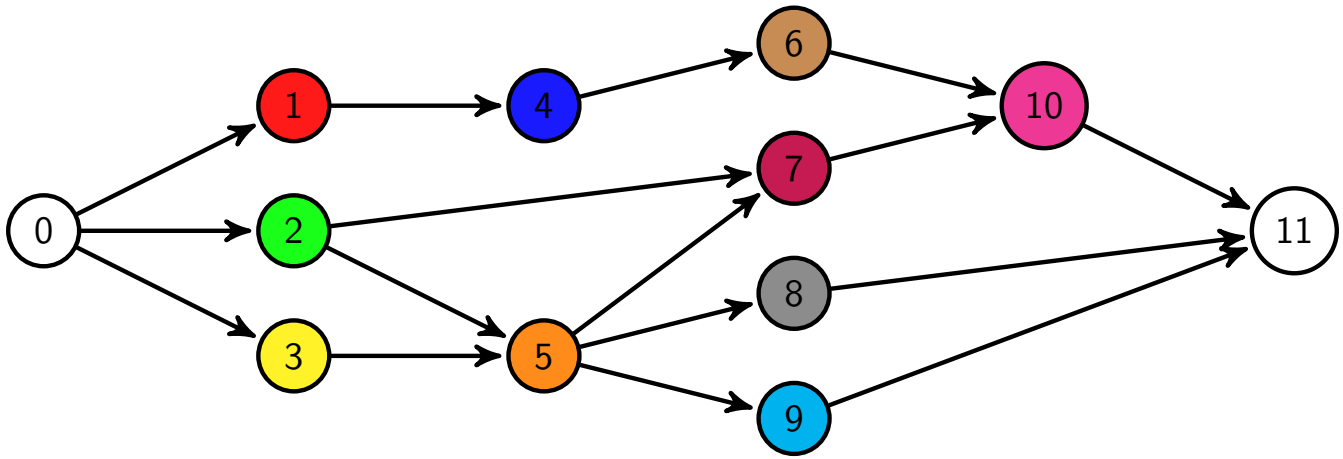
$$
\begin{aligned}
\text{Min. } & S_{n+1} \\
\text{s. t. } & z_{ij} + z_{ji} \leq 1 \quad i,j \in A,\ i < j \\
& z_{ij} + z_{jh} - z_{ih} \leq 1 \quad i,j,h \in A,\ i \neq j \neq h) \\
& z_{ij} = 1 \quad (i,j) \in E \\
& S_j - S_i + M_{ij}(1 - z_{ij}) \geq p_i \quad i,j \in A,\ i \neq j \\
& \sum_{i,j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

[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

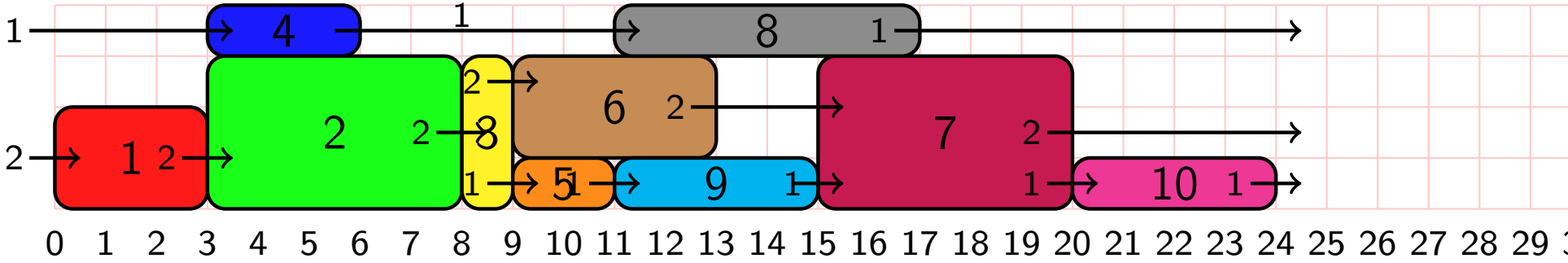
$\phi_{ij}^k \geq 0$: nombre d'unités de la ressource k transférées de i à j

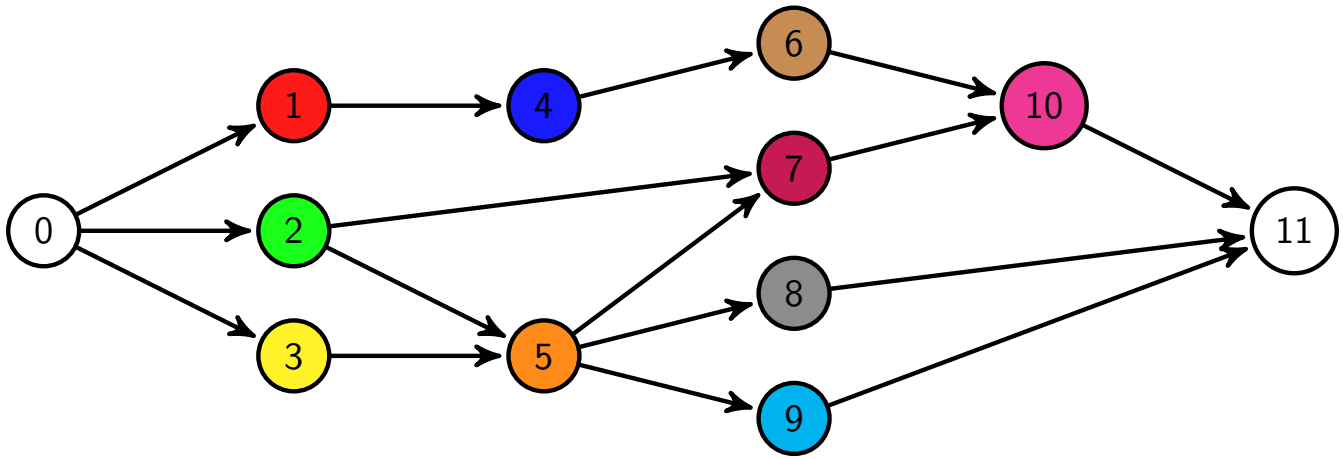




Le concept de flot d'unités de ressources

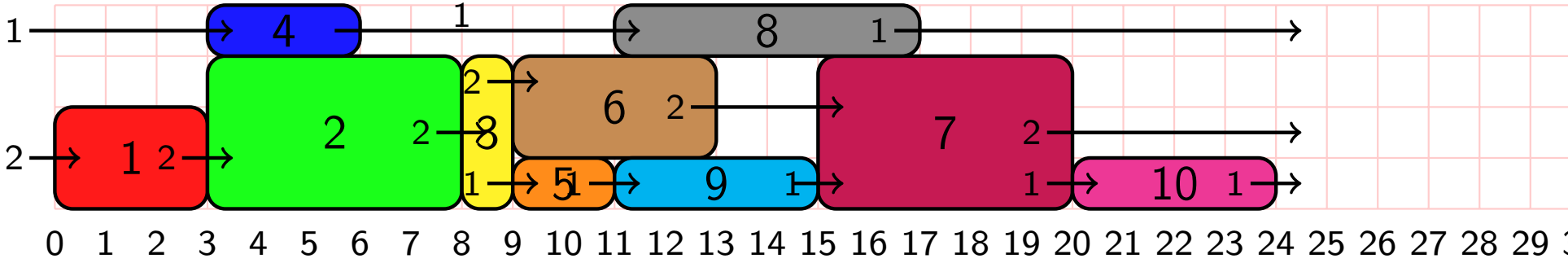
$\phi_{ij}^k \geq 0$: nombre d'unités de la ressource k transférées de i à j



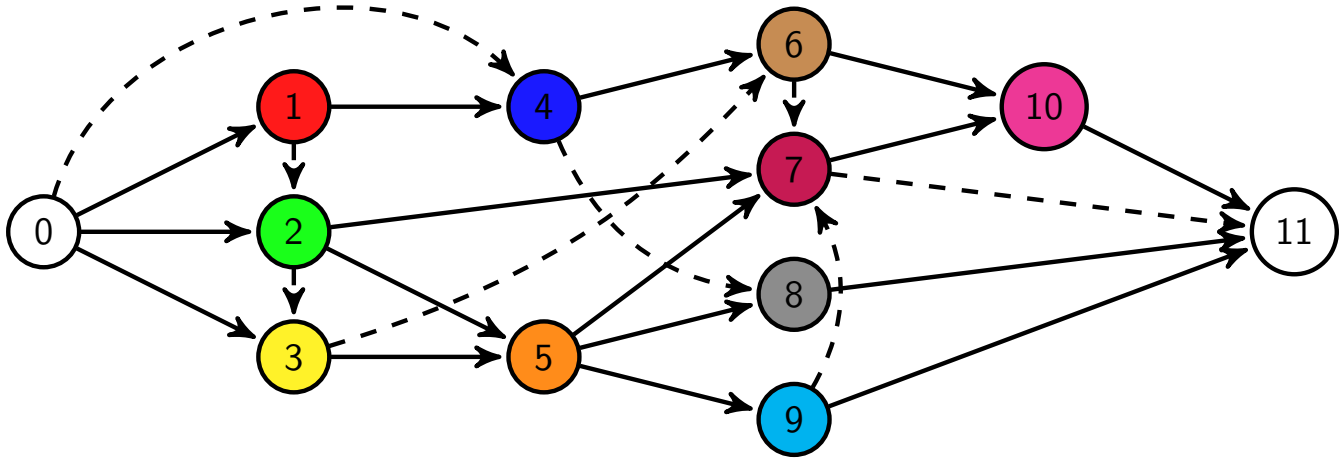


Le concept de flot d'unités de ressources

$\phi_{ij}^k \geq 0$: 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 :

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

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

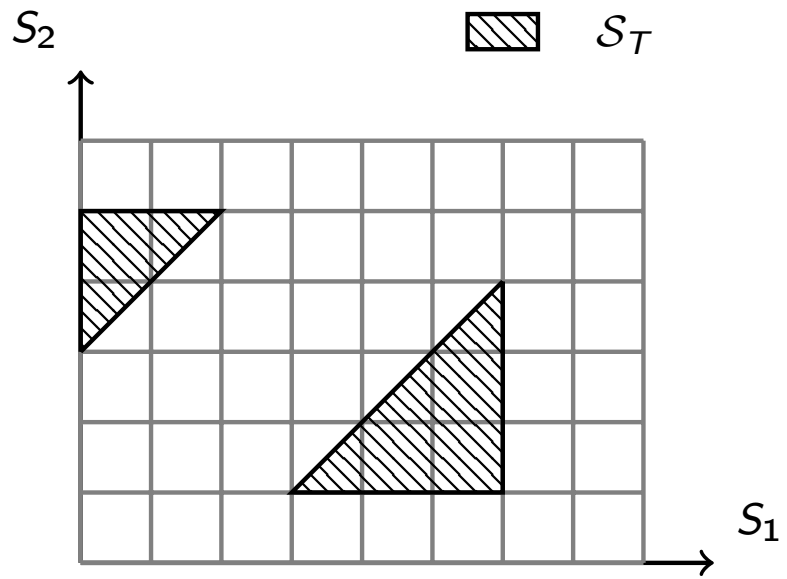
$$\sum_{i \in V \setminus \{j\}} \phi_{ij}^k = \tilde{r}_{jk} \quad (j \in V \setminus \{0\})$$

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

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

Relaxation de mauvaise qualité

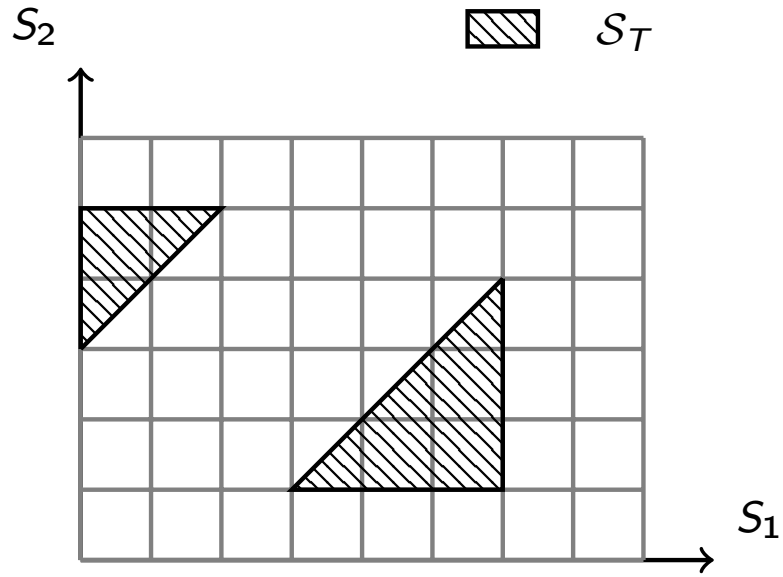
- Exemple 2 tâches, $ES = (0, 1), LS = (6, 5)$, un ensemble critique $\{1, 2\}$, minimisation de la somme des dates de fin



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

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

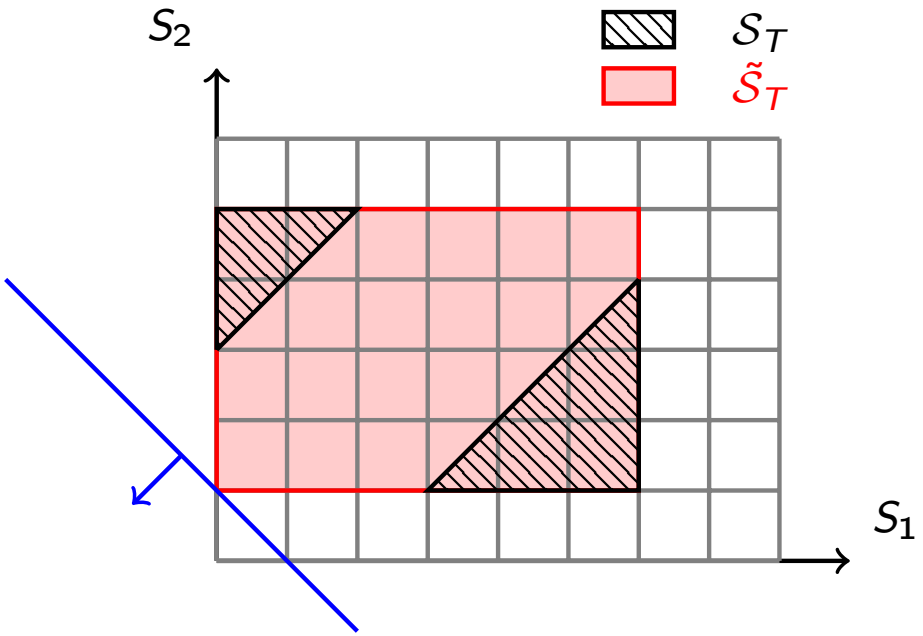


L'ensemble des solutions réalisable du PLNE est bien $\mathcal{S}_T$

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

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

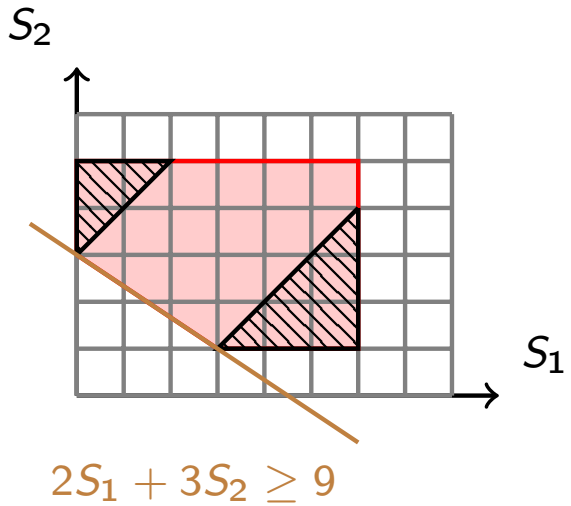


relaxation LB=6

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...)



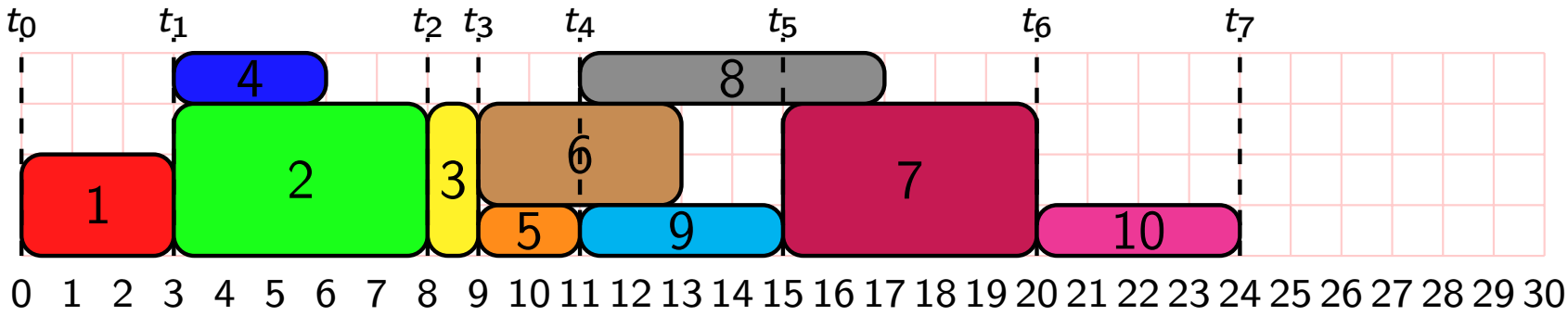
$$(p_i + ES_i - ES_j)S_i + (o_j + ES_j - ES_i) \geq p_i p_j + ES_i p_j + ES_j p_i$$

- 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 1994,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

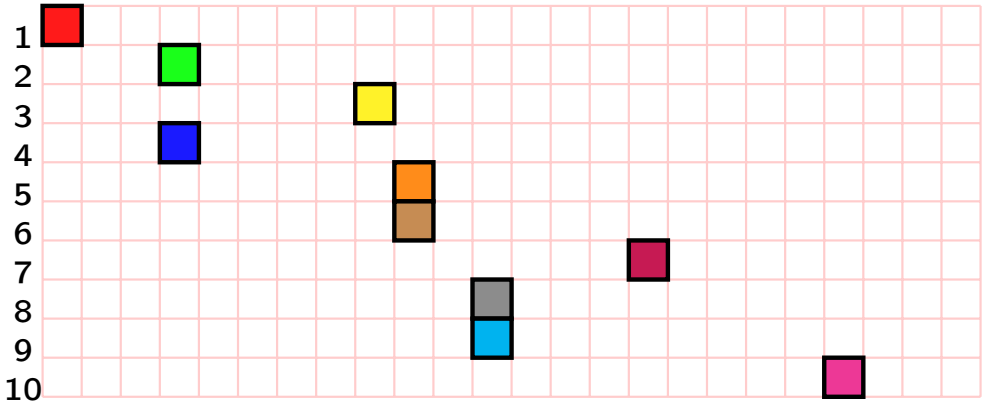
- Relaxation également de mauvaise qualité pour le RCPSP [Koné *et al.*. 2011]
- Inégalités valides sur les problèmes de minimisation de la somme des dates de fins sur une machine [Della croce *et al.* 2014]
- Inégalités valides sur le modèle on-off [Nattaf, Kis, A., Lopez 2016]
  - Contraintes de non 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)

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

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

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

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

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

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

[Pritsker et al. 1969]

Retour au petit exemple, une meilleure relaxation...

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

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

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

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

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

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

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

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

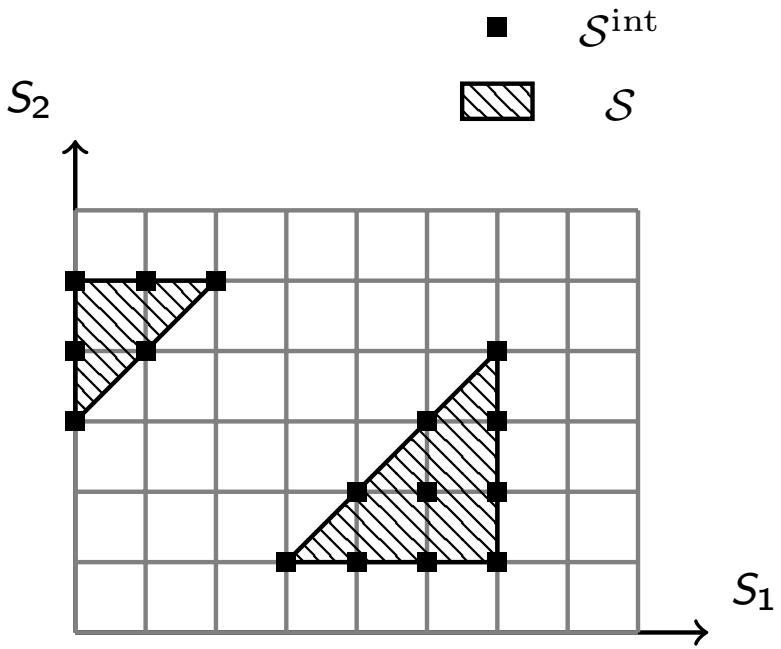
$$x_{2,3} + x_{2,4} + x_{1,2} + x_{1,3} + x_{1,4} \leq 1$$

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

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

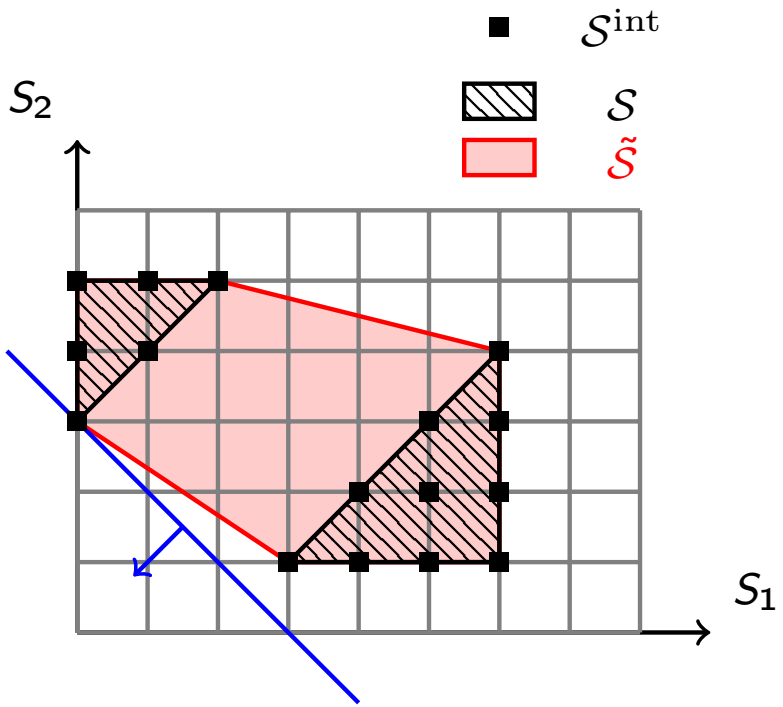
$$x_{1,t} \in \{0,1\} \quad t \in \{0,\ldots,6\}$$

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



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...

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

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

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

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

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

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

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

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

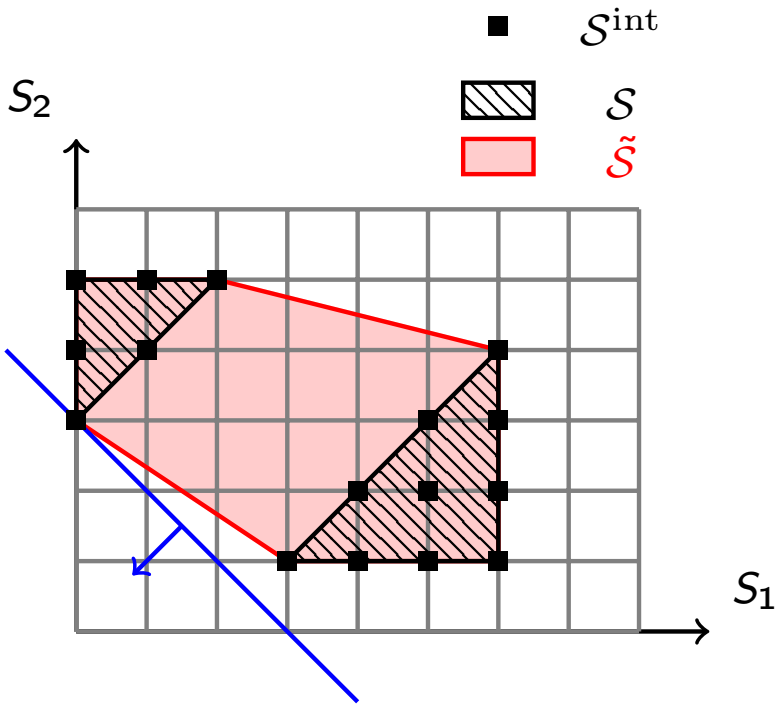
$$x_{2,3} + x_{2,4} + x_{1,2} + x_{1,3} + x_{1,4} \leq 1$$

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

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

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

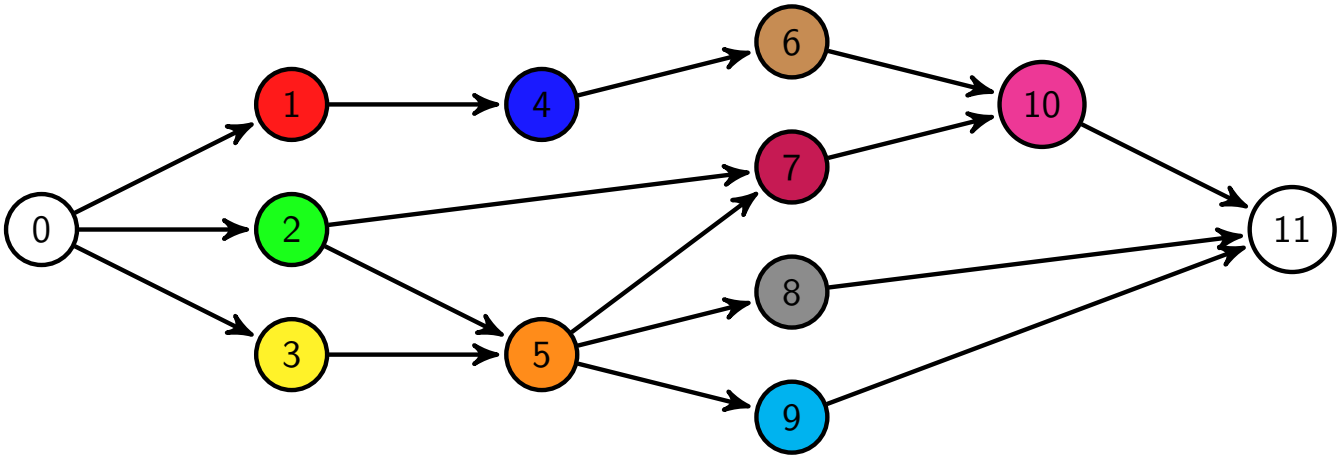
$$x_{2,t} \in \{0,1\} \quad t \in \{1,\ldots,5\}$$



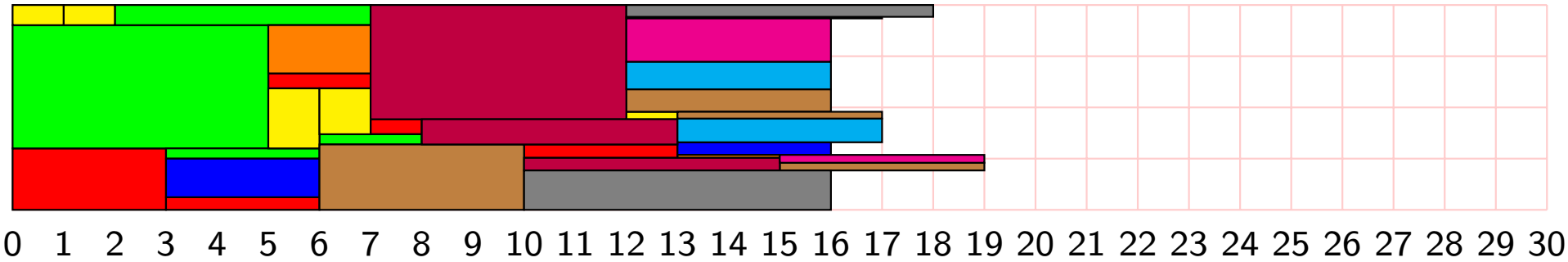
Dans cet exemple $\tilde{S}_T = conv(S_T)$, la relaxation est idéale.
... on a besoin de 11 variables binaires pour 2 tâches

... 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 :

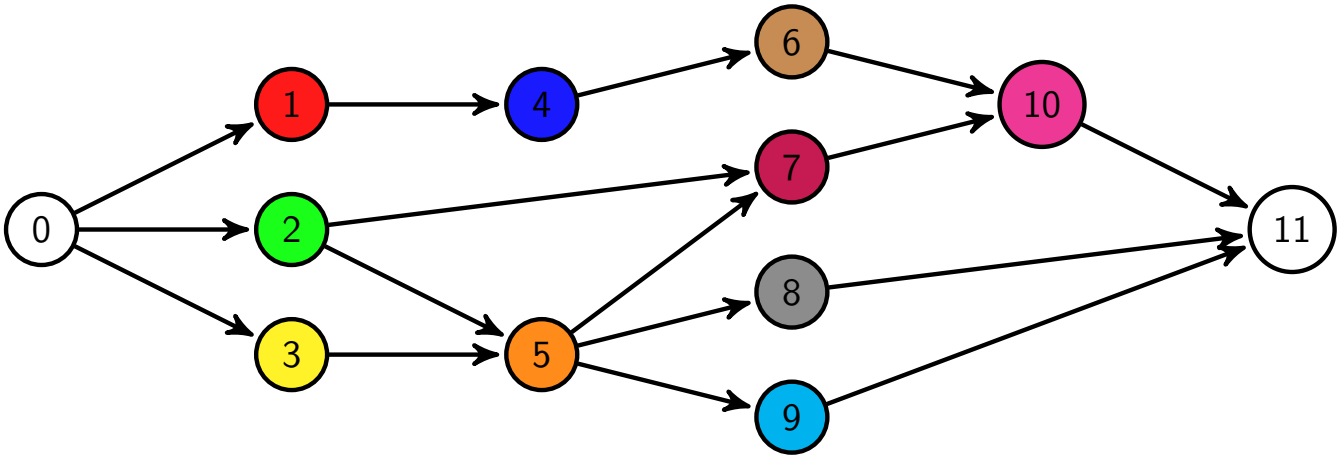
$$\sum_{\tau=0}^{t-p_i} x_{i\tau} - \sum_{\tau=0}^{t} x_{j\tau} \geq 0 \quad (i,j) \in E;\ t \in T$$

[Christofides *et al.* 1997]

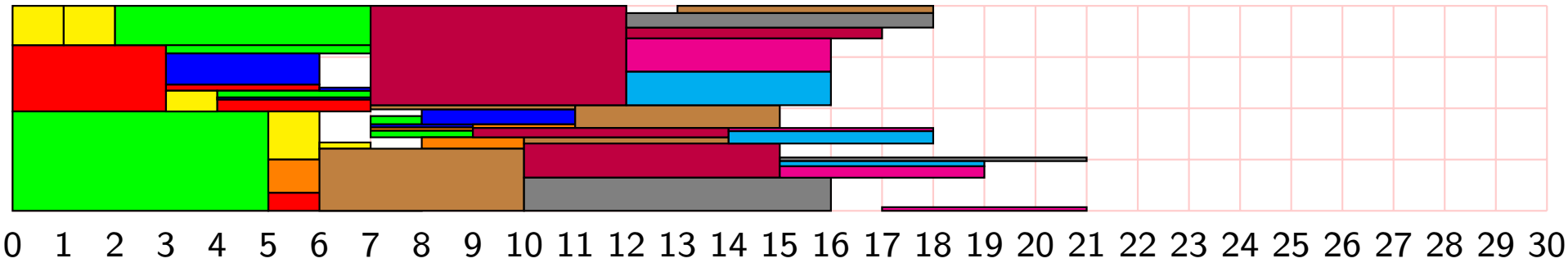
- Relation logique : $S_j \leq t \Rightarrow S_i \leq t - p_i$
- La matrice sans les contraintes de ressources est totalement unimodulaire (TU).
- La relaxation lagrangienne des contraintes de ressources préserve TU. Calcul efficace par un algo. de flot max [Möhring *et al.* 2003]

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. 2013], les formulations proposées par [Klein 2000], [Bianco and Caramia 2013] sont montrées équivalentes à ou moins fortes que (DDT)

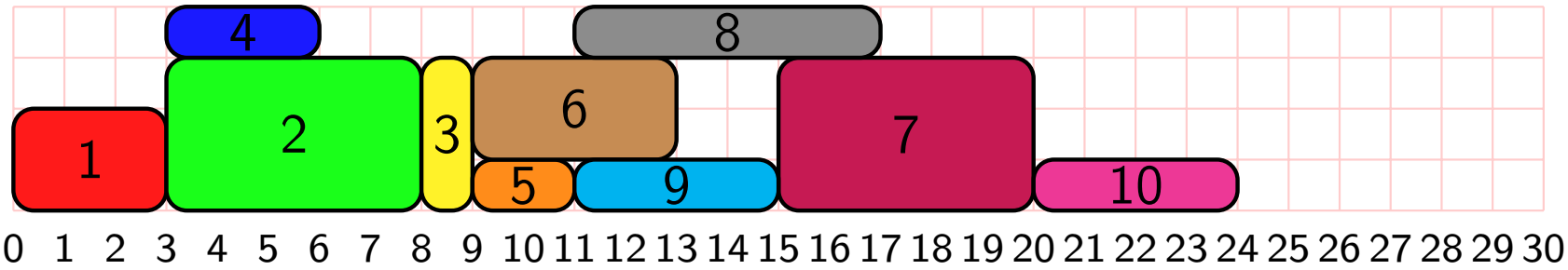
Inégalités valides

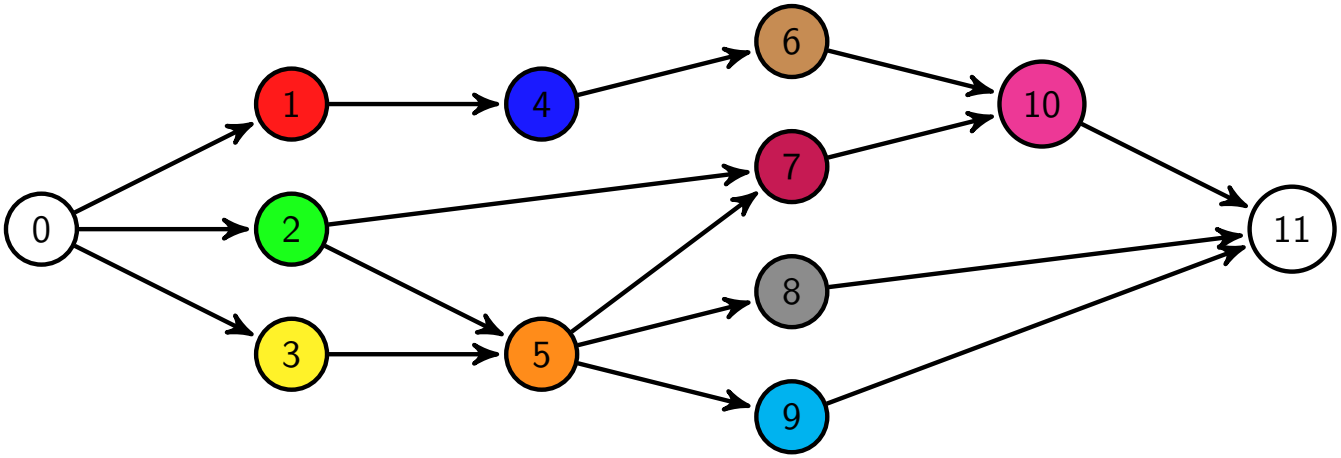
- Inégalités valides basées sur les ensembles critiques [Hardin *et al* 2008]
  - Basic inequality : $\sum_{i \in A} \sum_{s=t-p_i+1}^{t} x_{is} \leq |F| - 1, \quad \forall F \in \mathcal{F}$
  - Famille plus générale, extension aux intervalles de longueur v

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

  - 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]

1. Introduction : Les problèmes d'ordonnancement
2. Algorithmes (exacts) polynomiaux
3. Algorithmes de programmation dynamique
4. Algorithme de séparation et d'évaluation
5. Programmation linéaire en nombres entiers
6. Recherche arborescente dirigée par les conflits (Crédits : E. Hébrard)
   - Principe des solveurs SAT
   - Encodage SAT des problèmes d'ordonnancement
   - Méthodes hybrides PPC/SAT

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

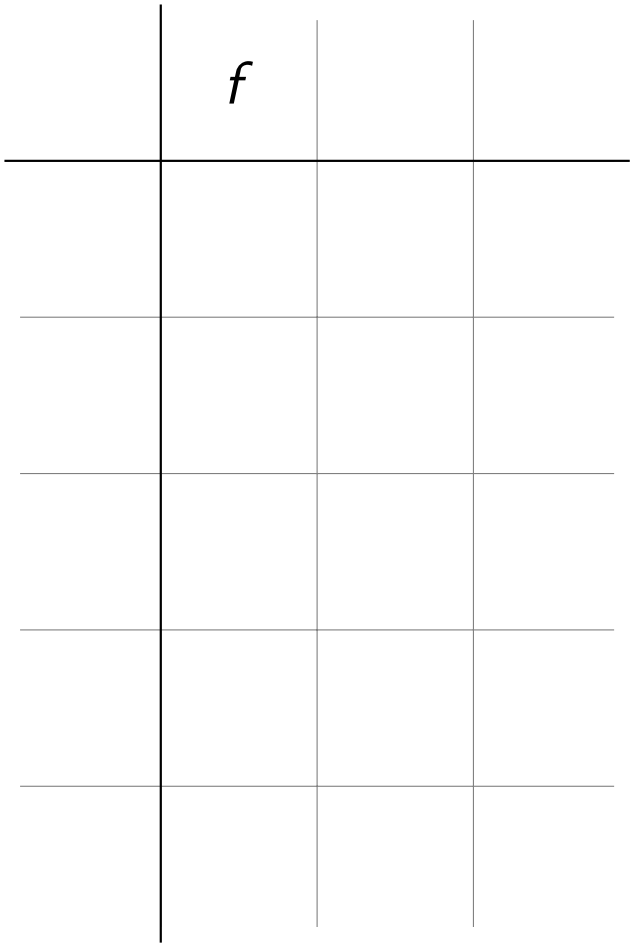
Apprentissage de clauses dirigé par les conflits (CDCL)

issu de DPLL

- apprentissage de clause (solveurs puis)
  - Premier 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.

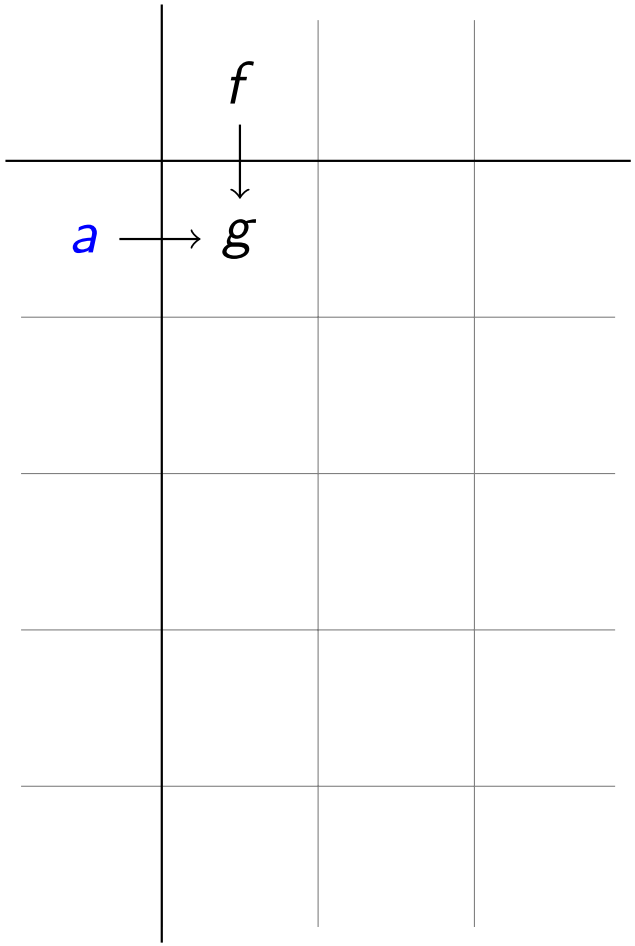
CDCL : Exemple



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

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

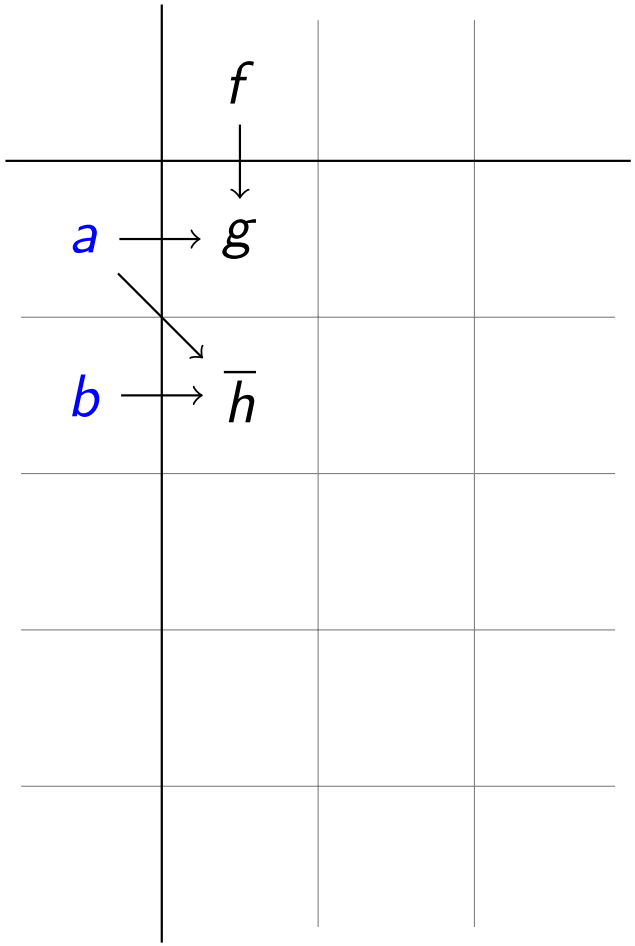
CDCL : Exemple



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

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

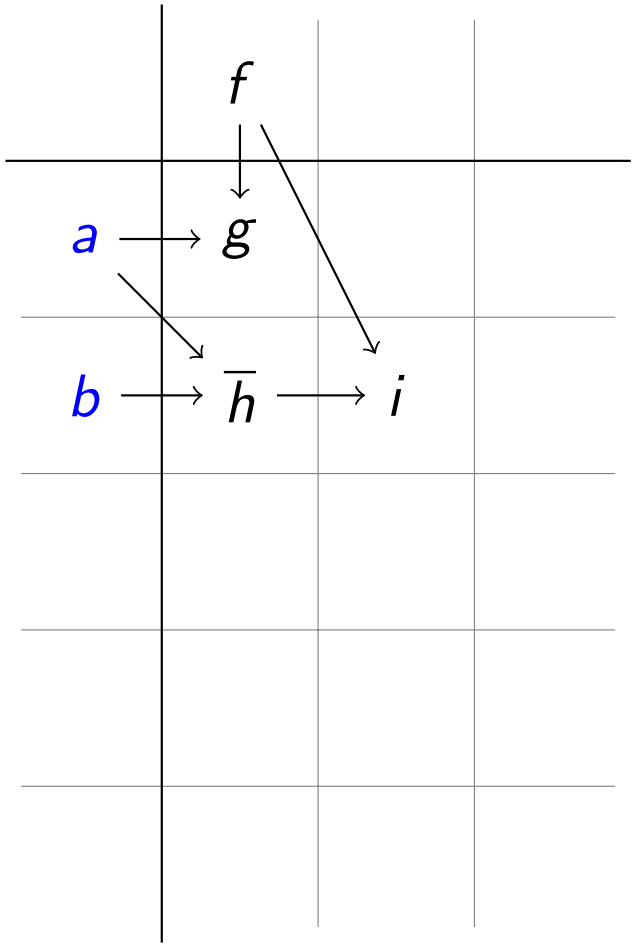
CDCL : Exemple



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

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

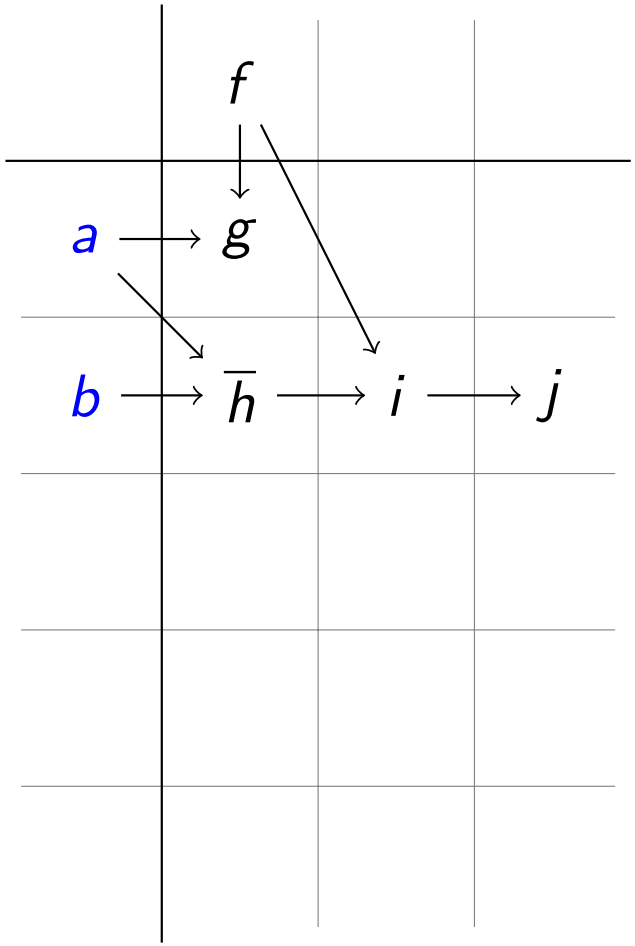
CDCL : Exemple



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

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

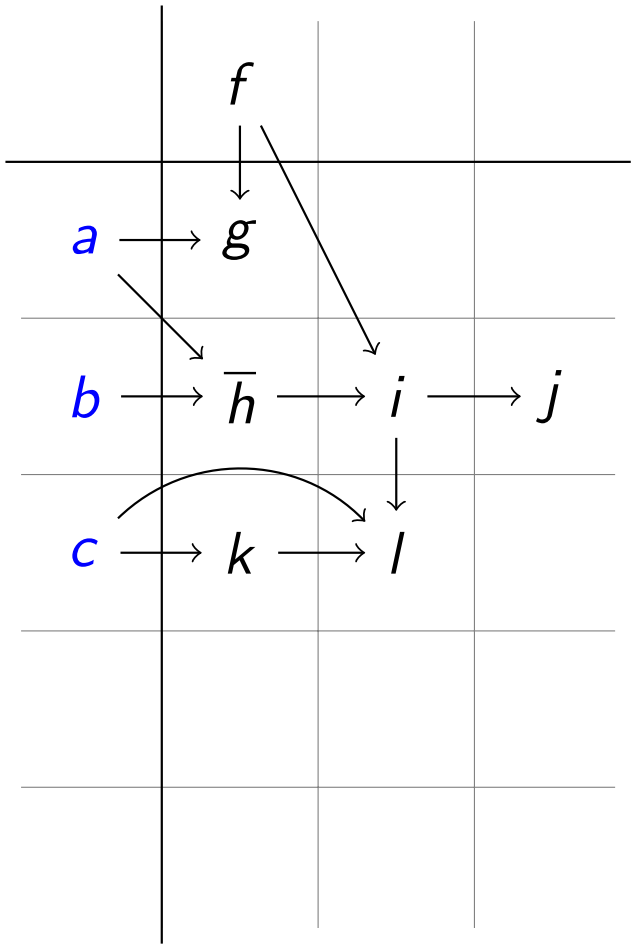
CDCL : Exemple



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

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

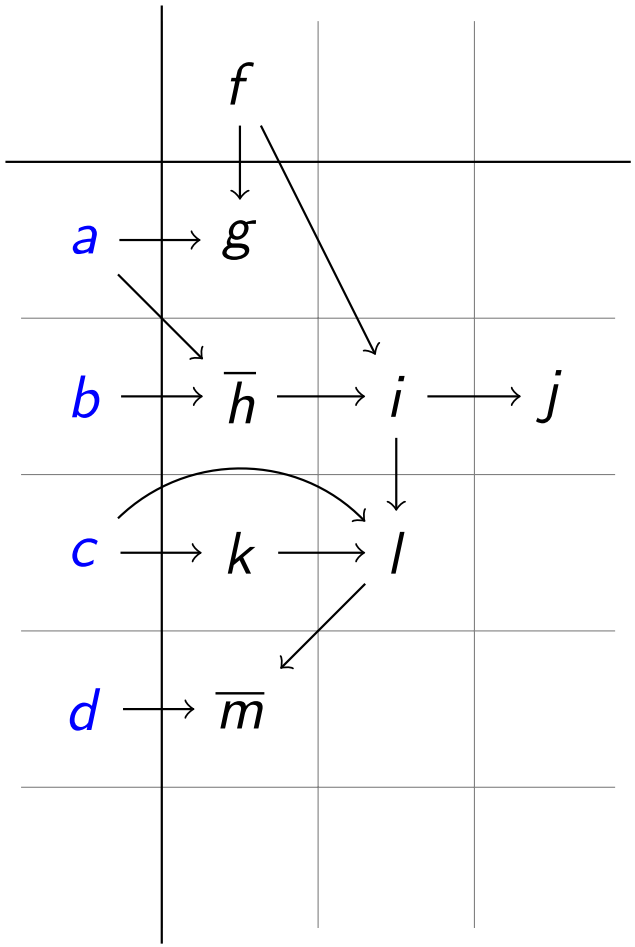
CDCL : Exemple



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

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

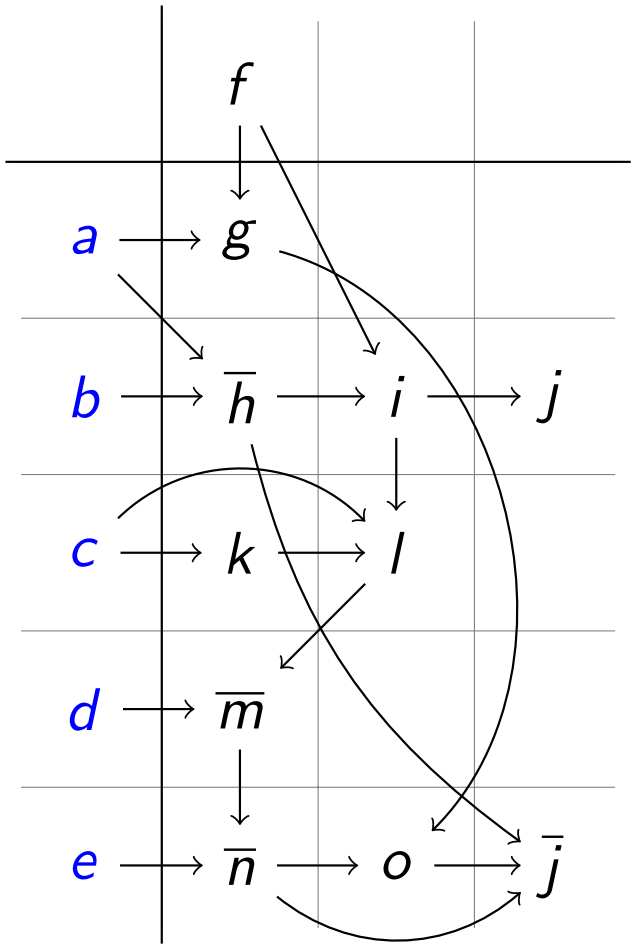
CDCL : Exemple



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

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

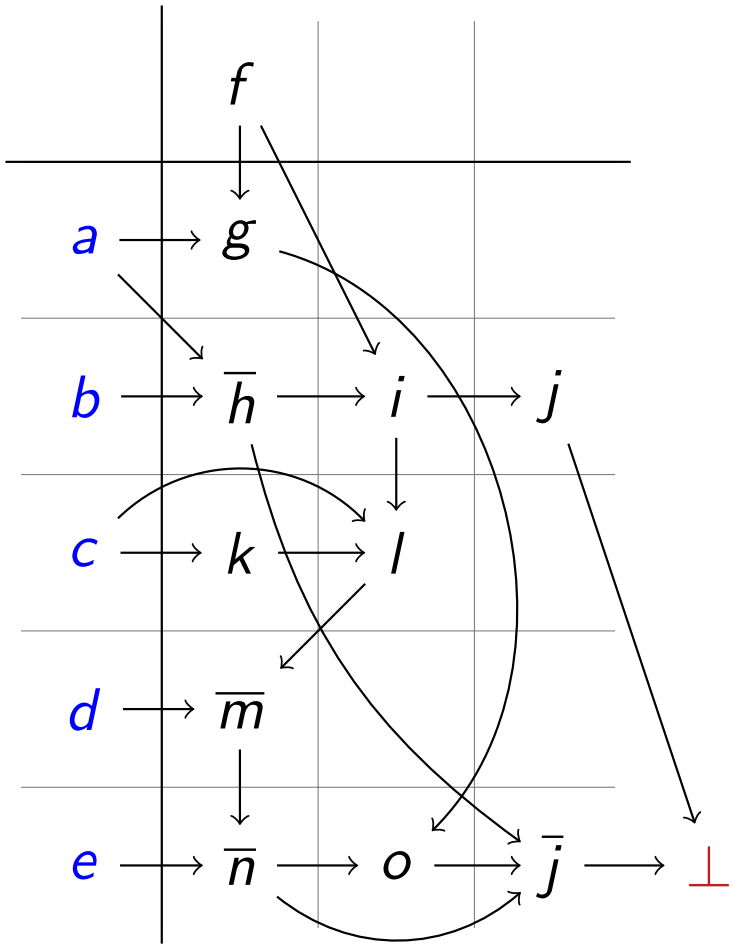
CDCL : Exemple



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

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

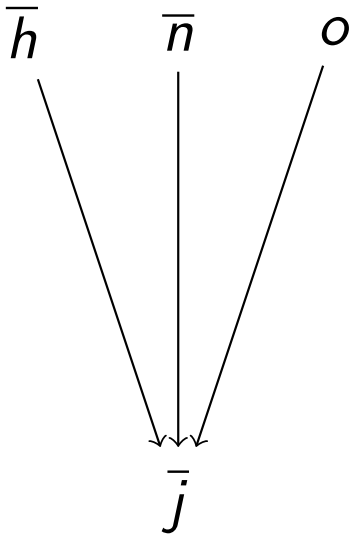
CDCL : Exemple



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

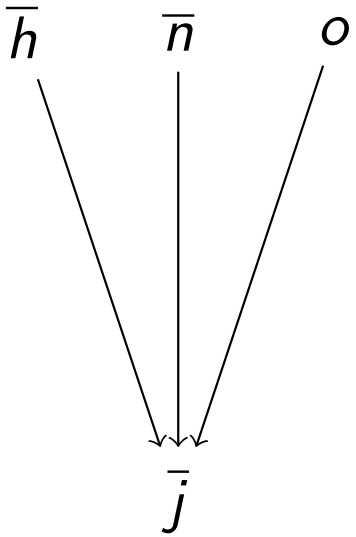
$c \vee h \vee n \vee \overline{m}$
$c \vee l$
$d \vee \overline{k} \vee l$
$d \vee \overline{g} \vee l$
$\overline{g} \vee n \vee o$
$\boxed{h \vee \overline{o} \vee \overline{j} \vee n}$
$\overline{i} \vee j$
$\overline{d} \vee \overline{l} \vee \overline{m}$
$\overline{e} \vee m \vee \overline{n}$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

CDCL : Example



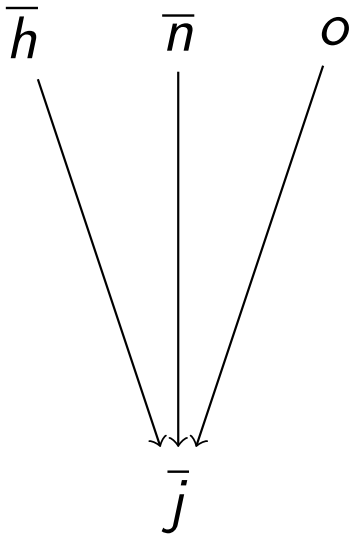
$$(h \vee \overline{o} \vee \overline{j} \vee n)$$

CDCL : Example



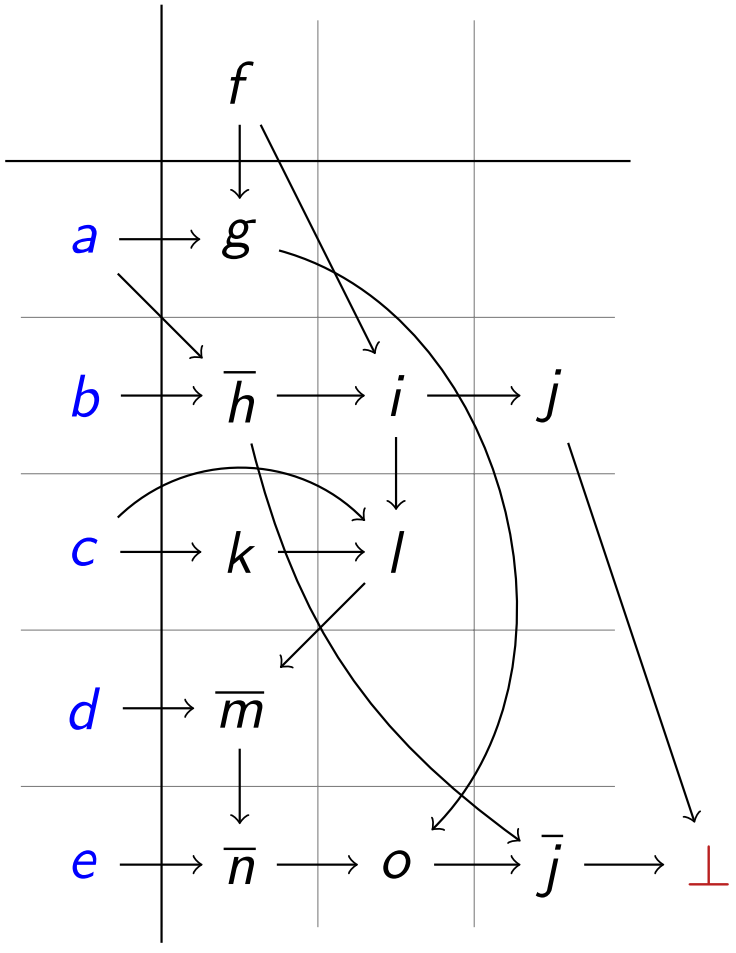
$$(h \vee \overline{o} \vee \overline{j} \vee n)$$

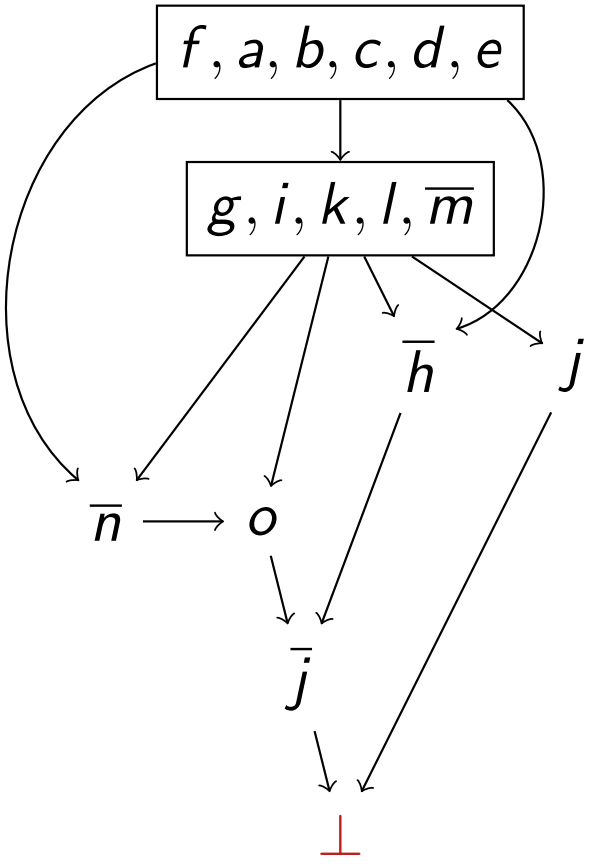
CDCL : Example



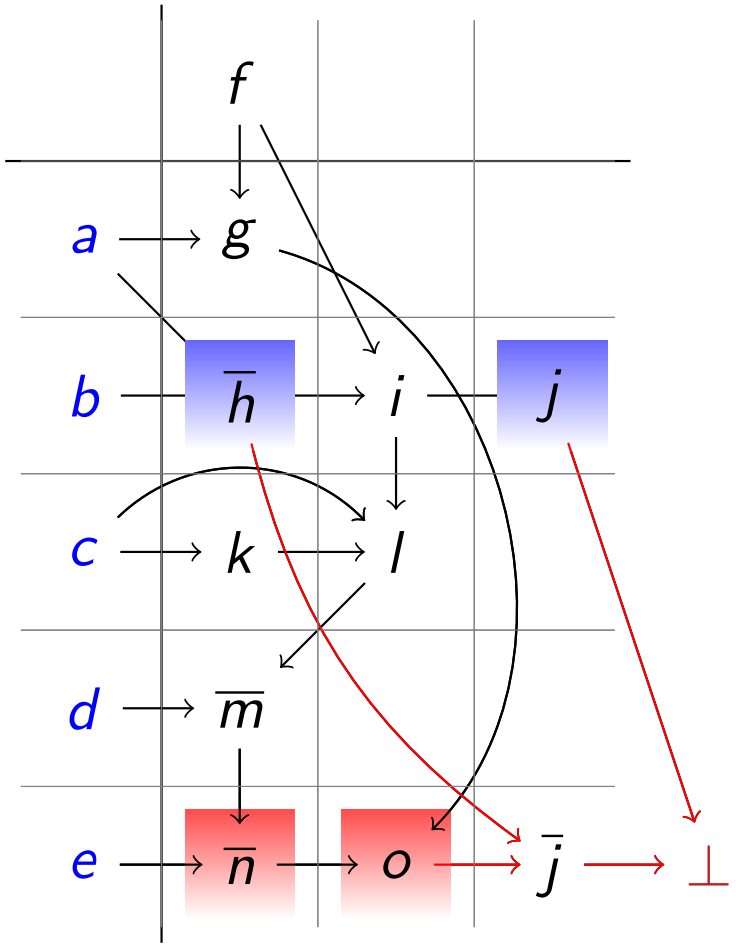
$$(h \vee \overline{o} \vee \overline{j} \vee n)$$
$$\equiv$$
$$(\overline{h} \wedge o \wedge \overline{n}) \rightarrow \overline{j}$$

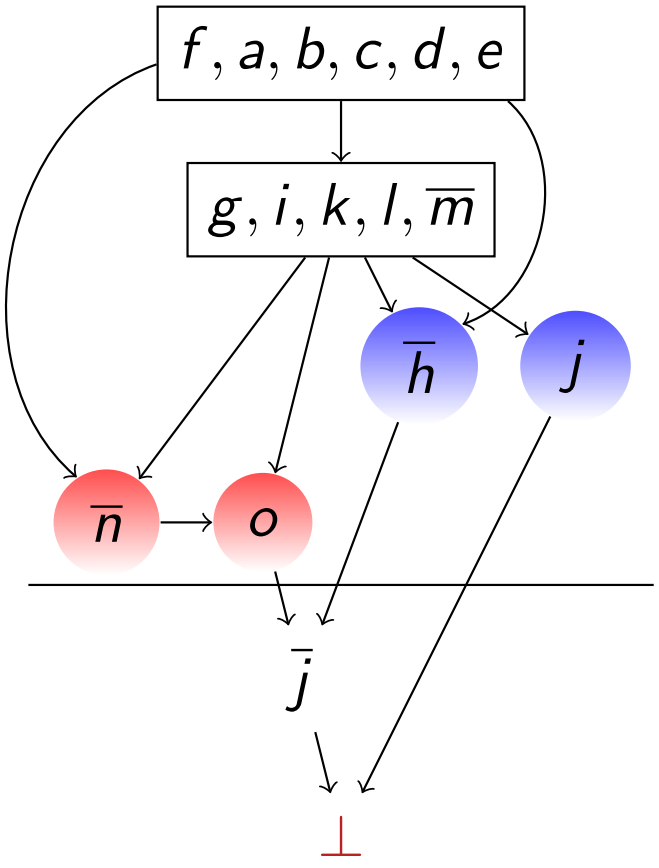
CDCL : Example



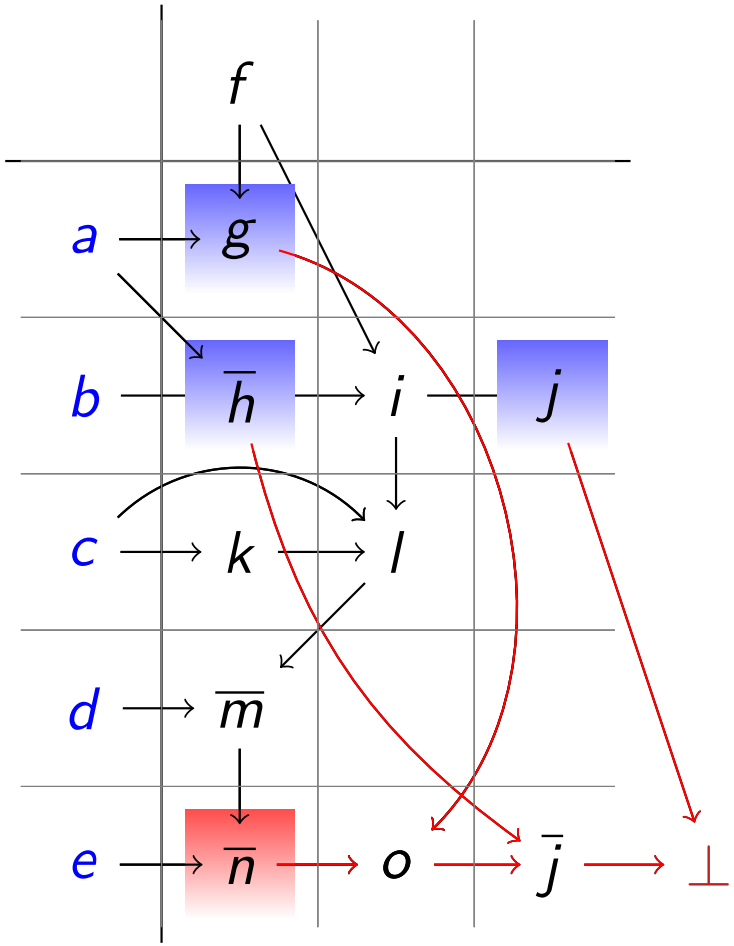


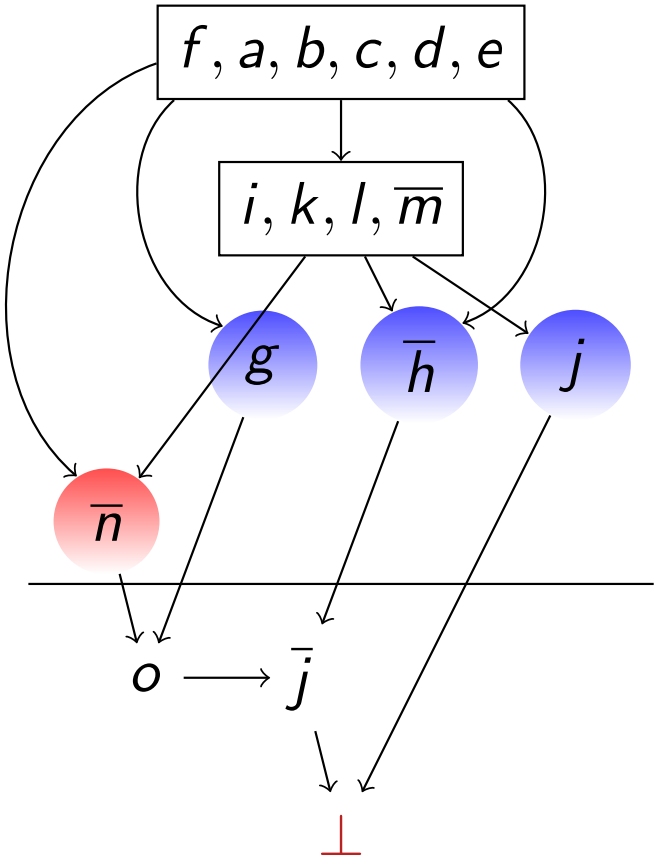
CDCL : Example



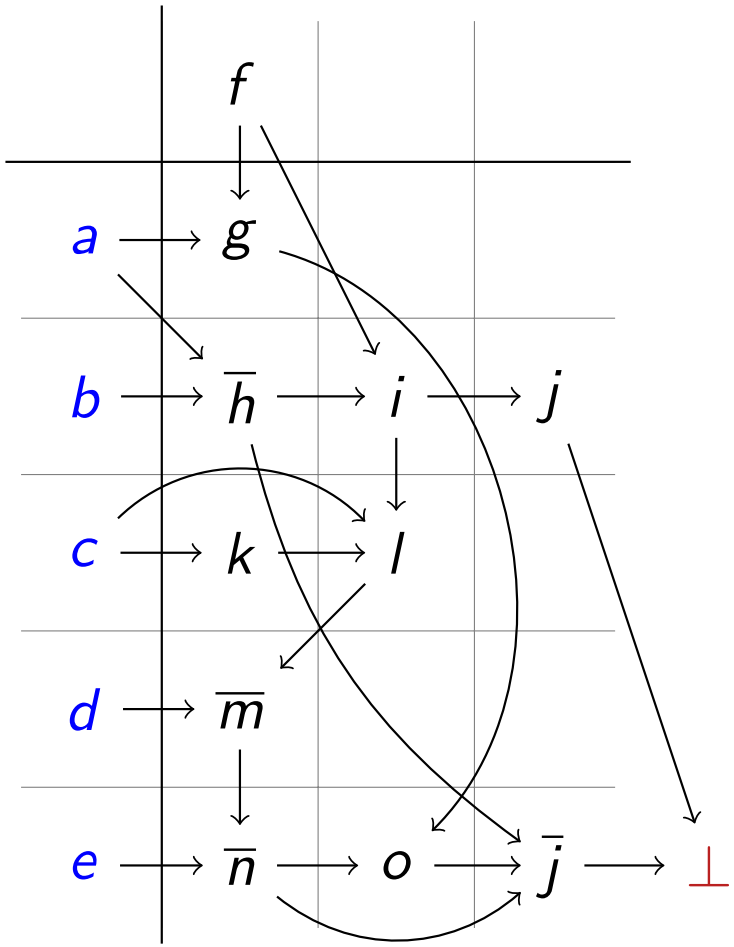


CDCL : Example





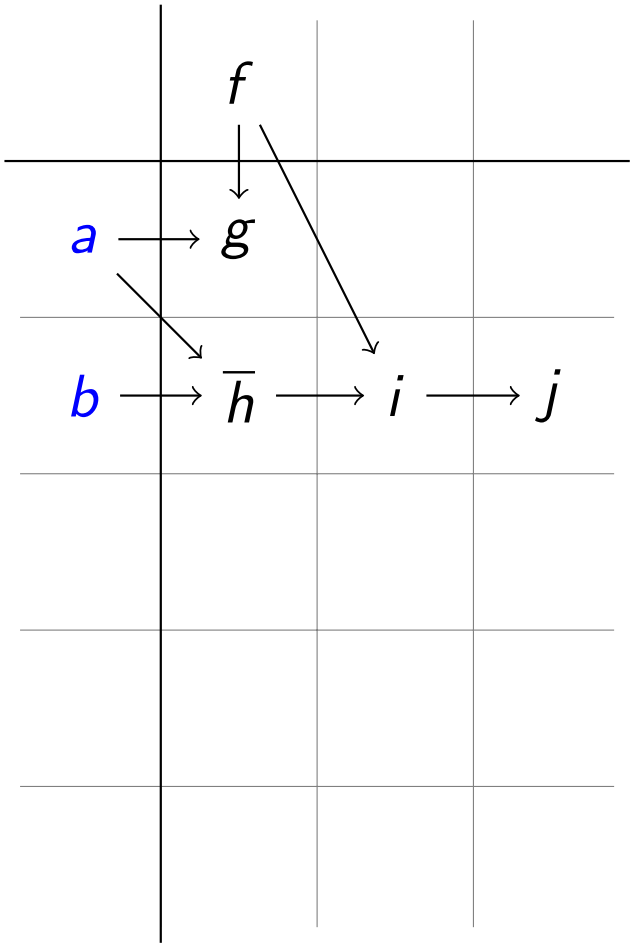
CDCL : Example



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

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

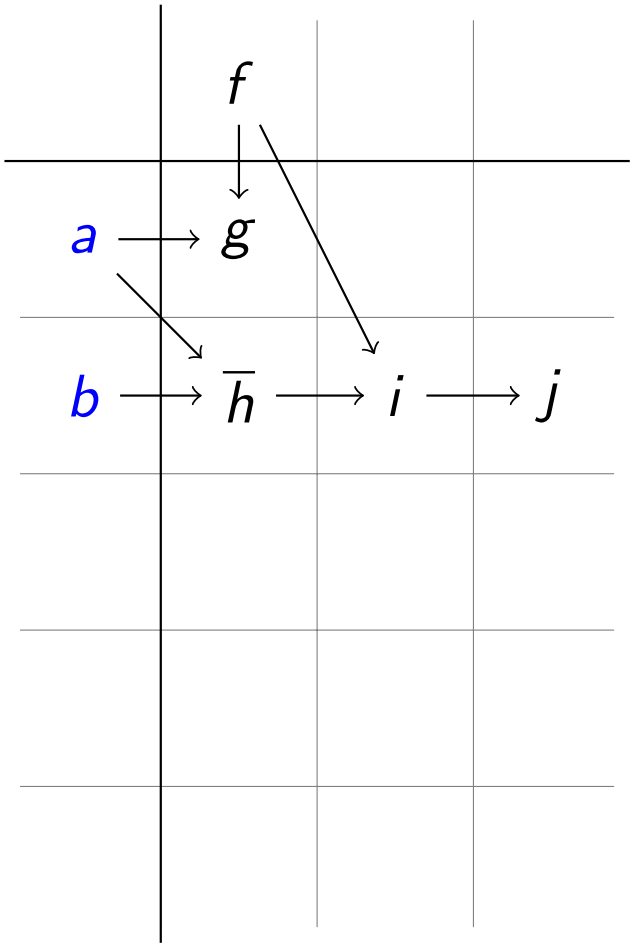
CDCL : Example



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

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

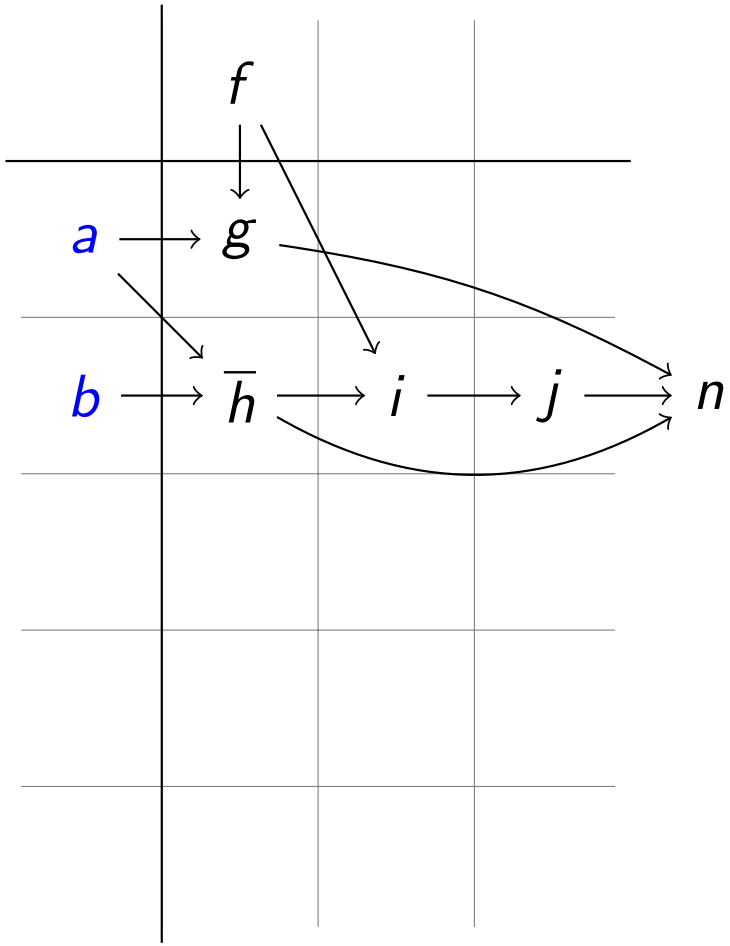
CDCL : Example



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

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

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 $(S_i, v \in [ES_i, LS_i])$
    - $i_v \Leftrightarrow x_i \leq v$

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

  - 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

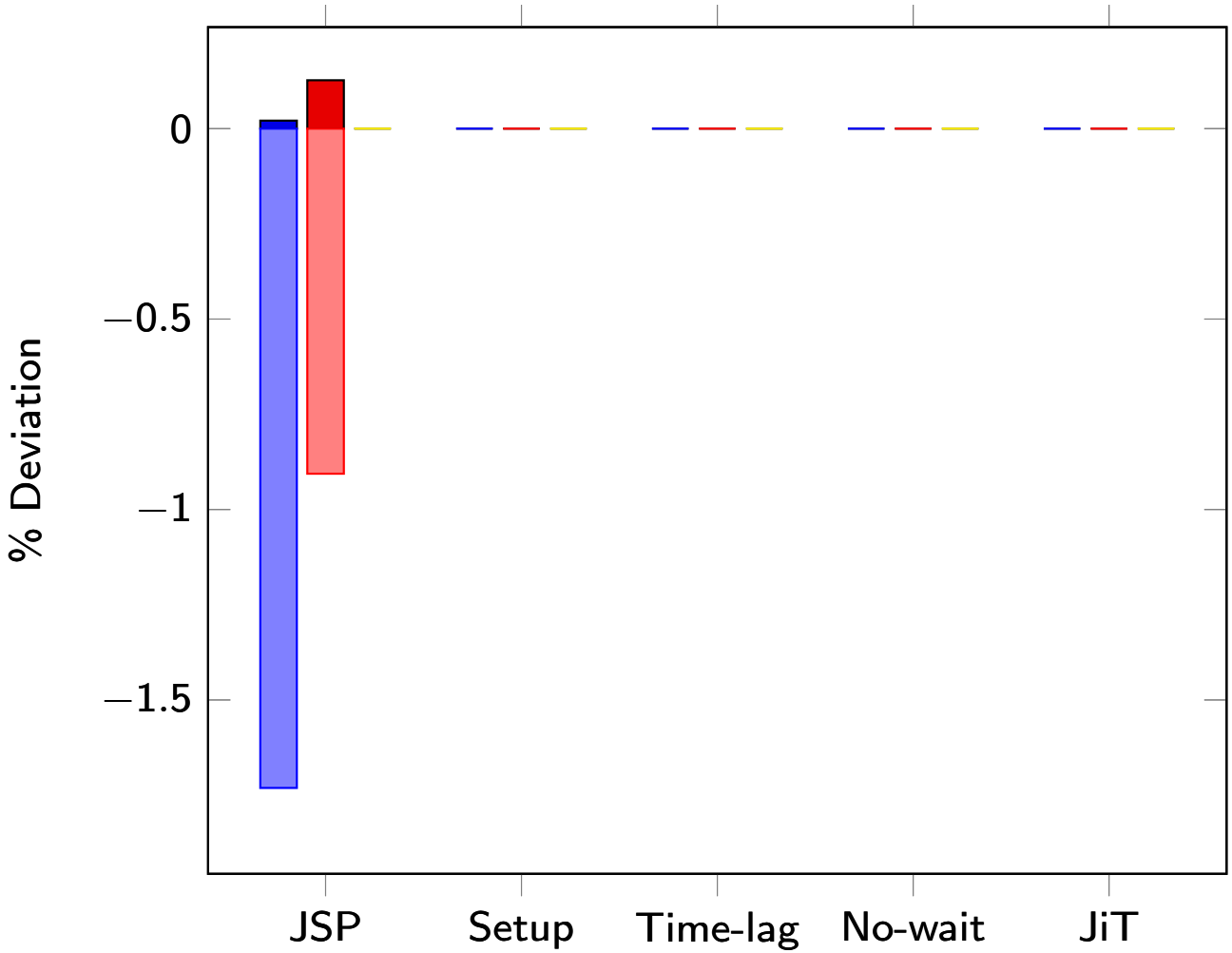
Resultat de l'encodage SAT

- Encodage plus efficace : mélange d'Order Encoding et de "Log Encoding"
- [Tamura, Tanjo & Banbara 2006] : solveur (Sugar) capable de résoudre toutes les instances ouvertes d'open shop.
- Encodage SAT $\rightarrow$ parfois une méthode efficace !

Résultat d'approches PPC "à la SAT"

- variables : dates de début S_i et variables booléennes $x_{ij} = 0$ si $S_i \geq S_j + p_j$ et $x_{ij} = 1$ si $S_j \geq S_i + p_j$
- [Grimes, Hébrard 2015] : branch&Bound sur les variables x_{ij}
- Branchement : variable de plus petit raio taille du domaine / dégré pondéré
- Stratégie : guider la recherche par la meilleure solution connue (phase saving dans SAT)
- Redémarrages géométriques avec n-goods

Jobshop - C_{max} - Taillard

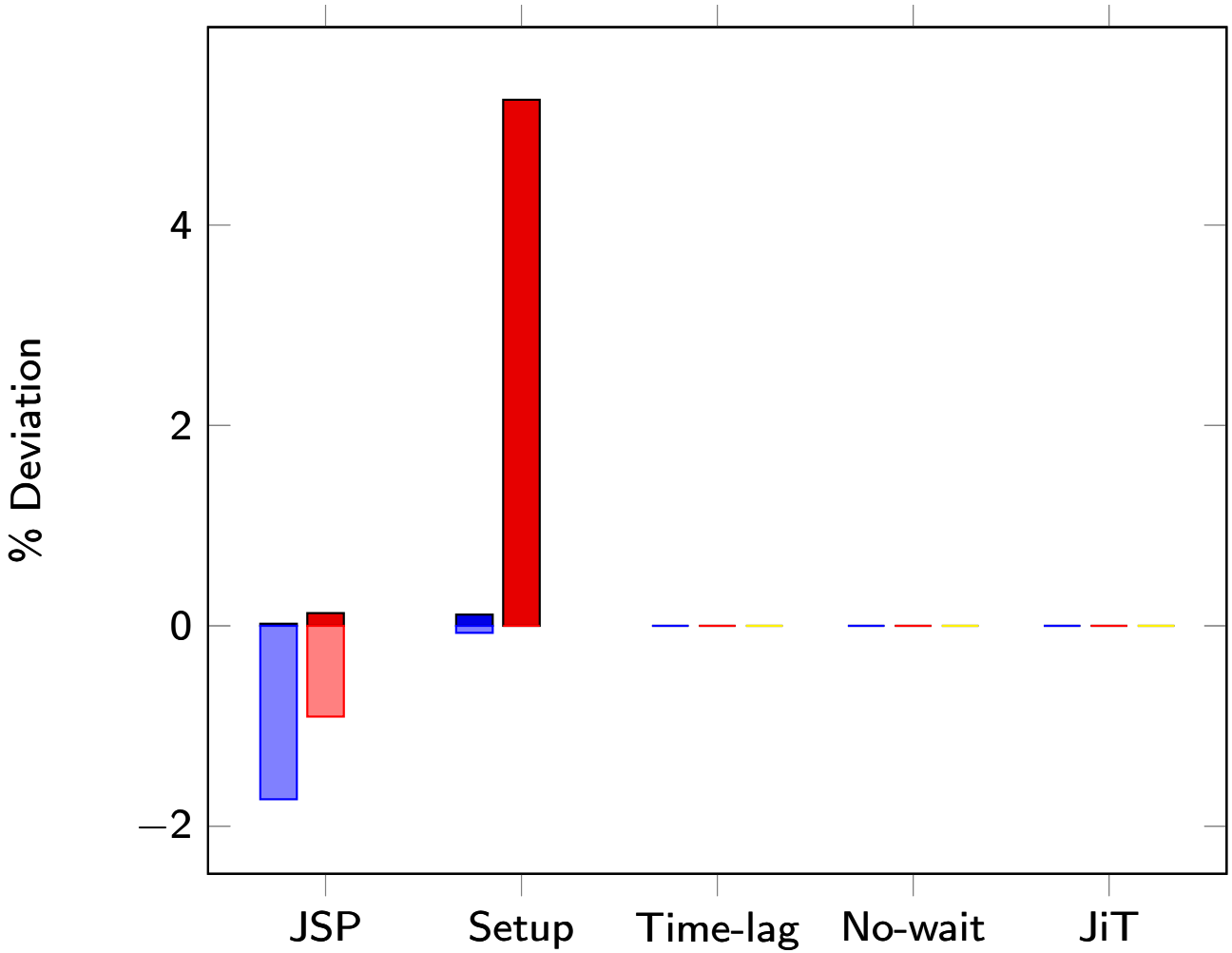


LS [1]
CP [2]
MIP (no data)

[1] i-TSAB (Tabu Search)
E. Nowicki and C. Smutnicki
J. of Scheduling (2005)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

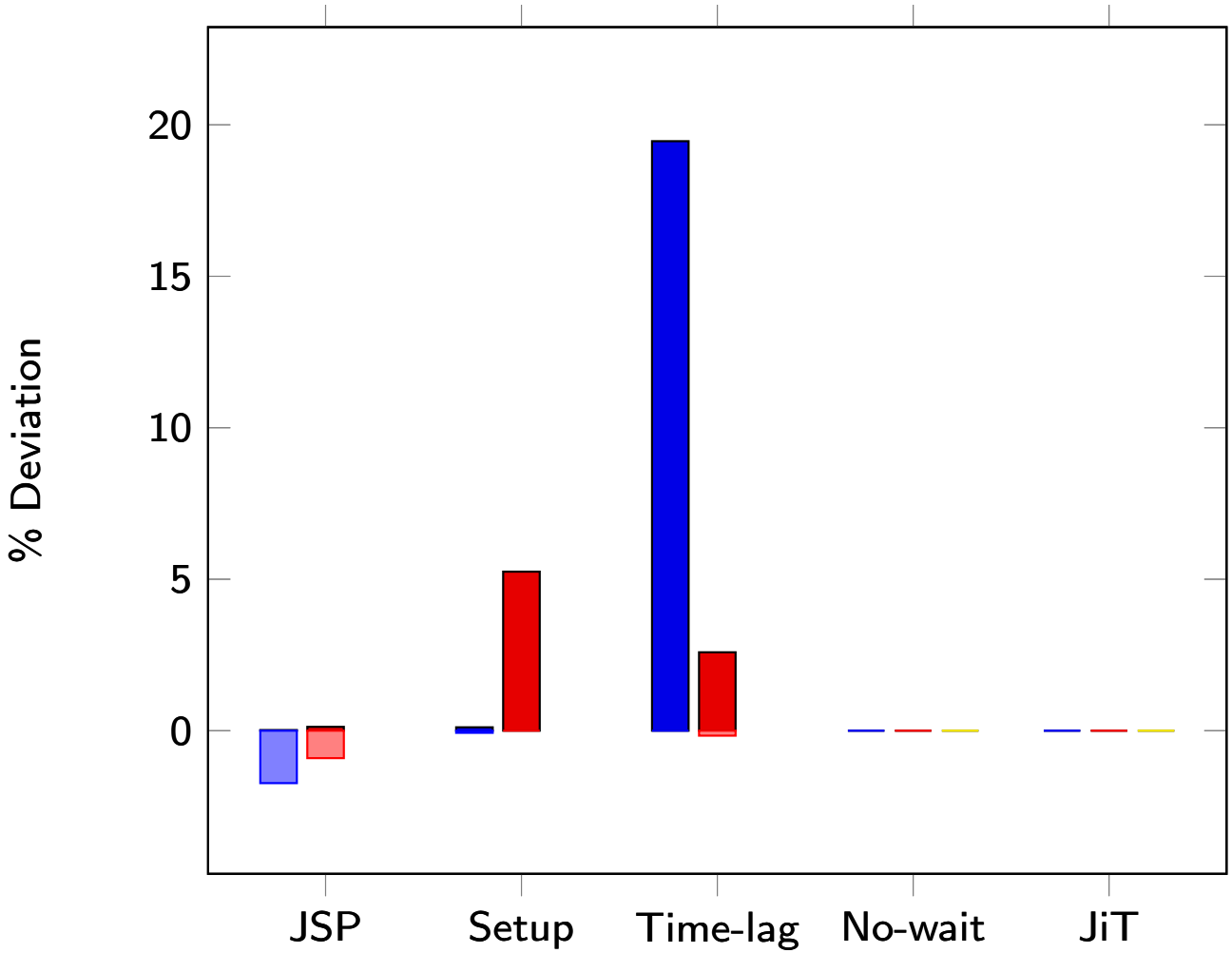
Jobshop with setup times - C_{max} - Brucker & Thiele



[3] GA / Tabu Search
Gonzalez, Vela, and Varela
ICAPS (2008)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

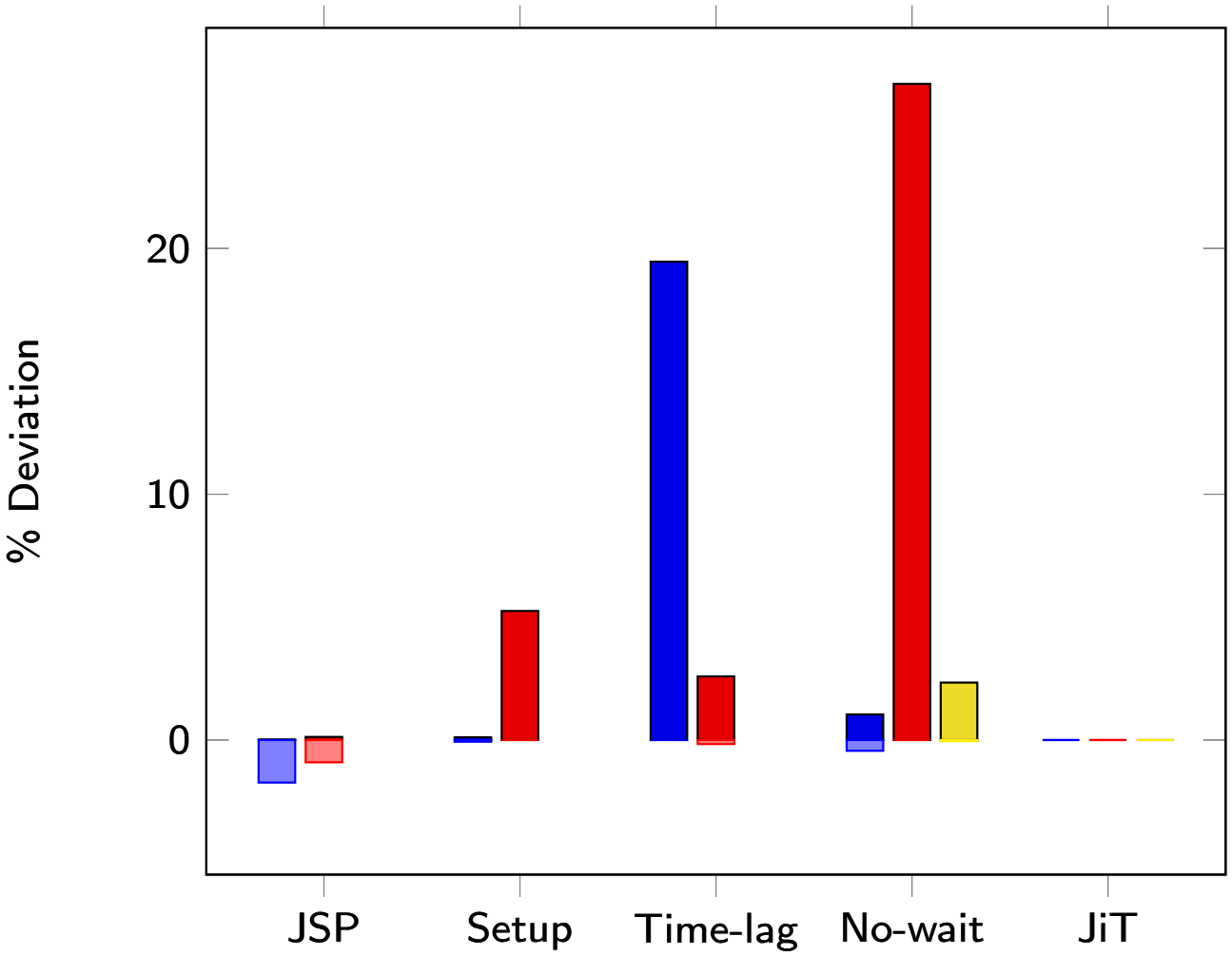
Jobshop with time lags - C_{max} - Lawrence (modified)



[4] Memetic algorithm
Caumond, Lacomme and Tchernev
Computers & OR (2008)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

"No-wait" Jobshop - C_{max} - Lawrence

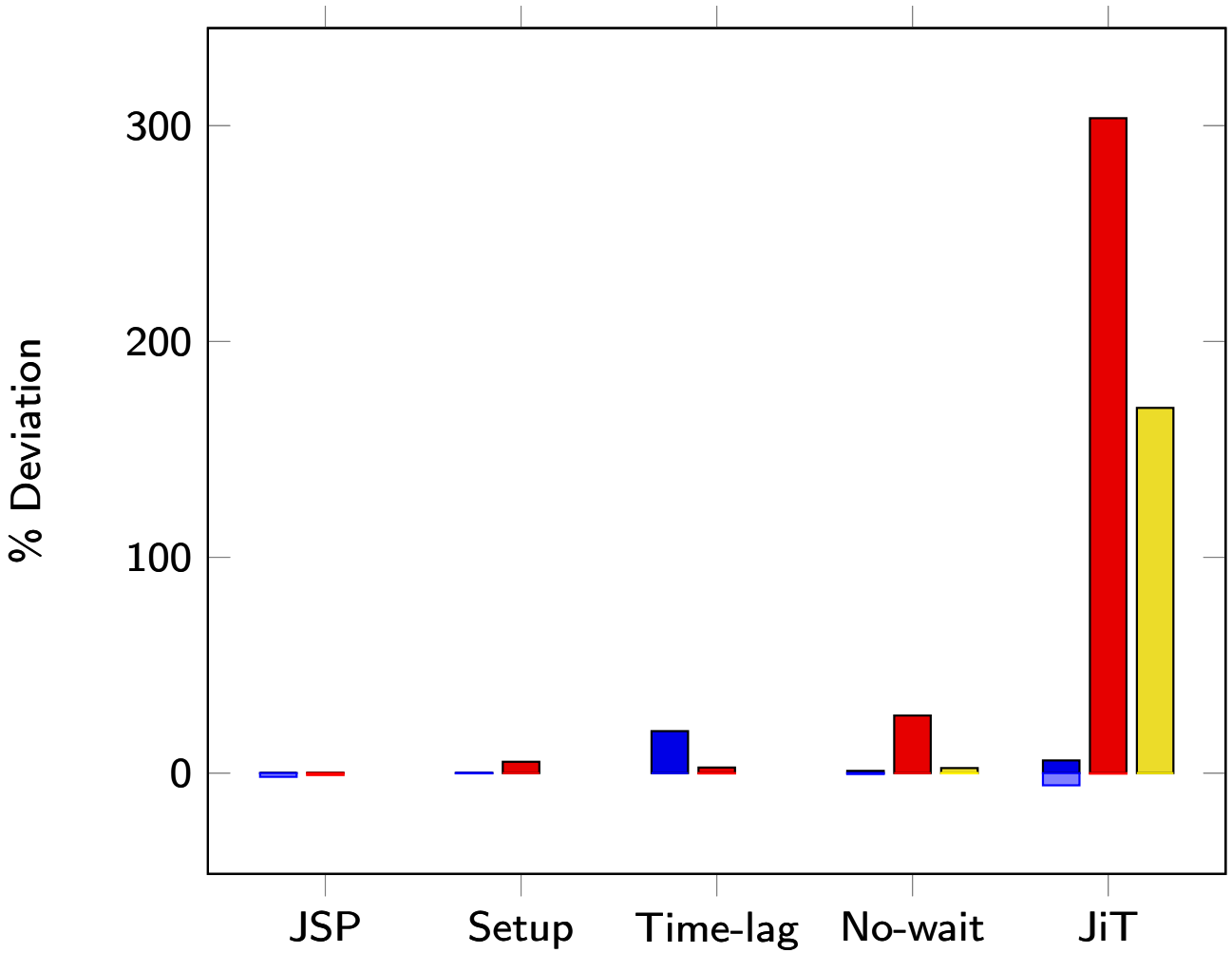


[5] Hybrid Tabu Search
Bozejko and Makuchowski
Computers & IE (2009)

[2] SGMPCS (Ilog Scheduler)
J. C. Beck
JAIR (2007)

[6] CPlex
J. J. J. van den Broek
PhD thesis (2009)

Jobshop - earliness/tardiness - Beck & Refalo ; Morton & Pentico



LS [7]
CP [8]
MIP [9]

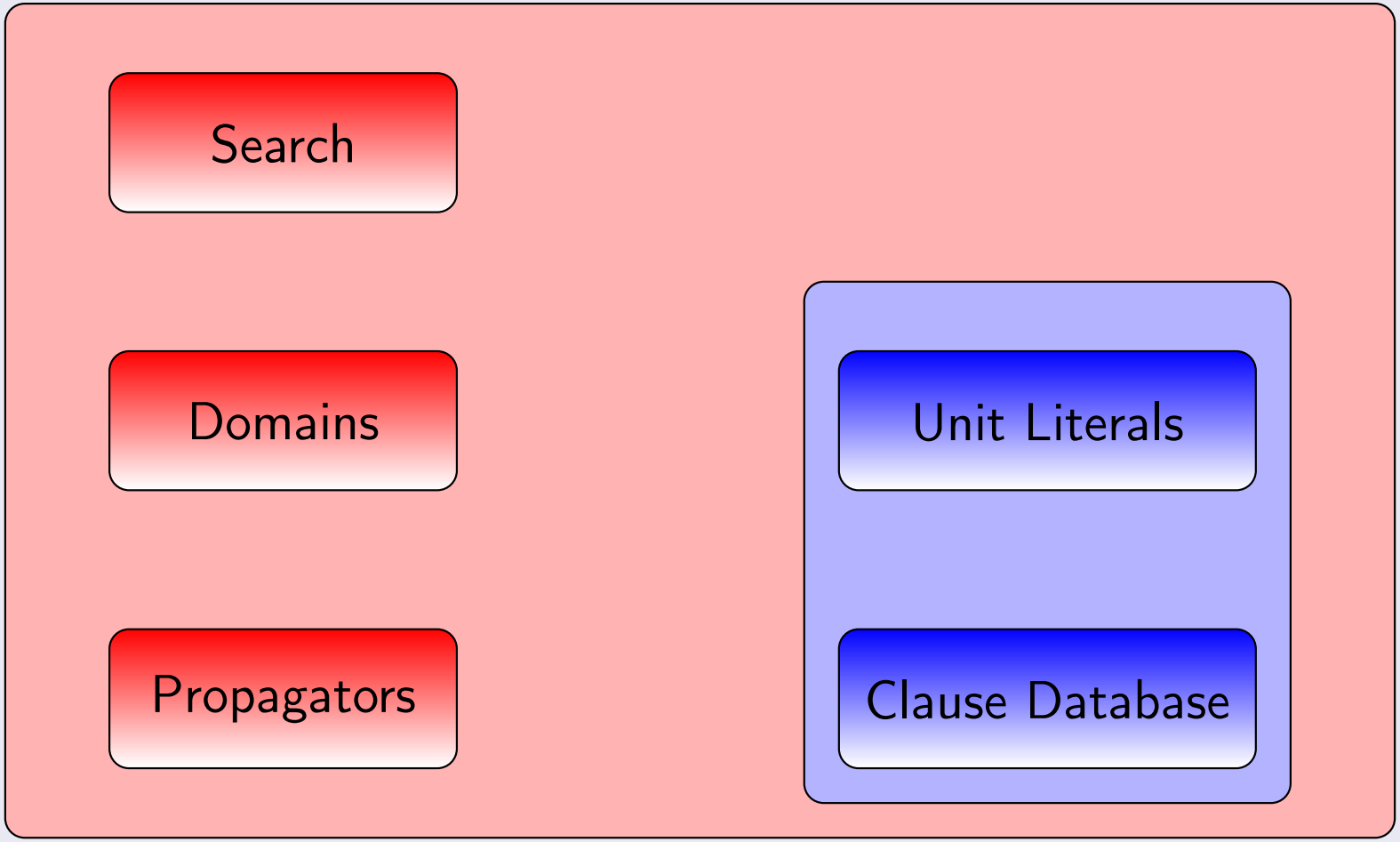
[7] LNS (CP/LP)
Danna and Peron
Ilog Tech. Report (2003)

[8] Hybrid CP/LP
Beck and Refalo
Annals of OR (2003)

[9] CPlex
Danna, Rothberg, Le Pape
CPAIOR (2003)

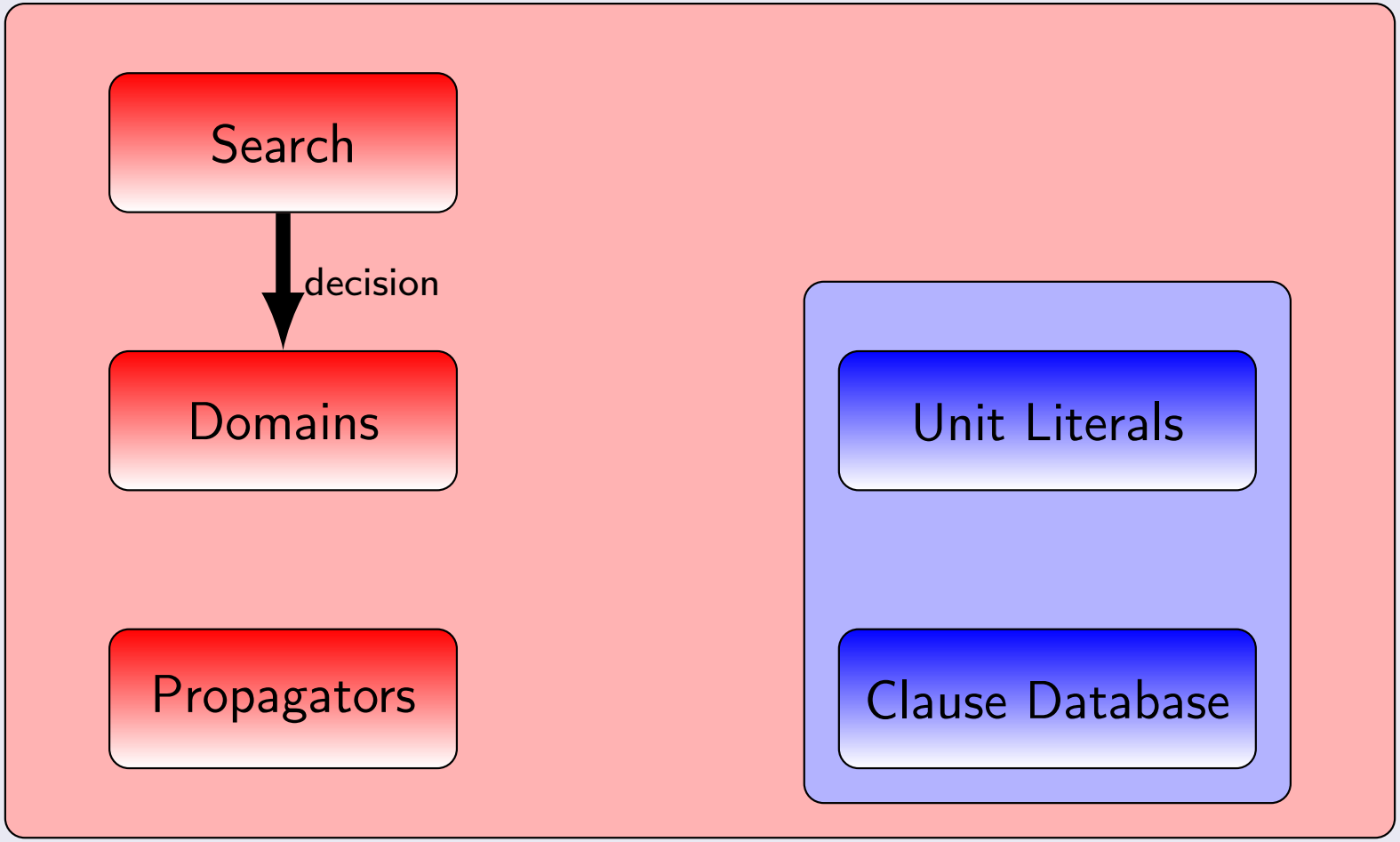
La génération de clauses retardée

Architecture



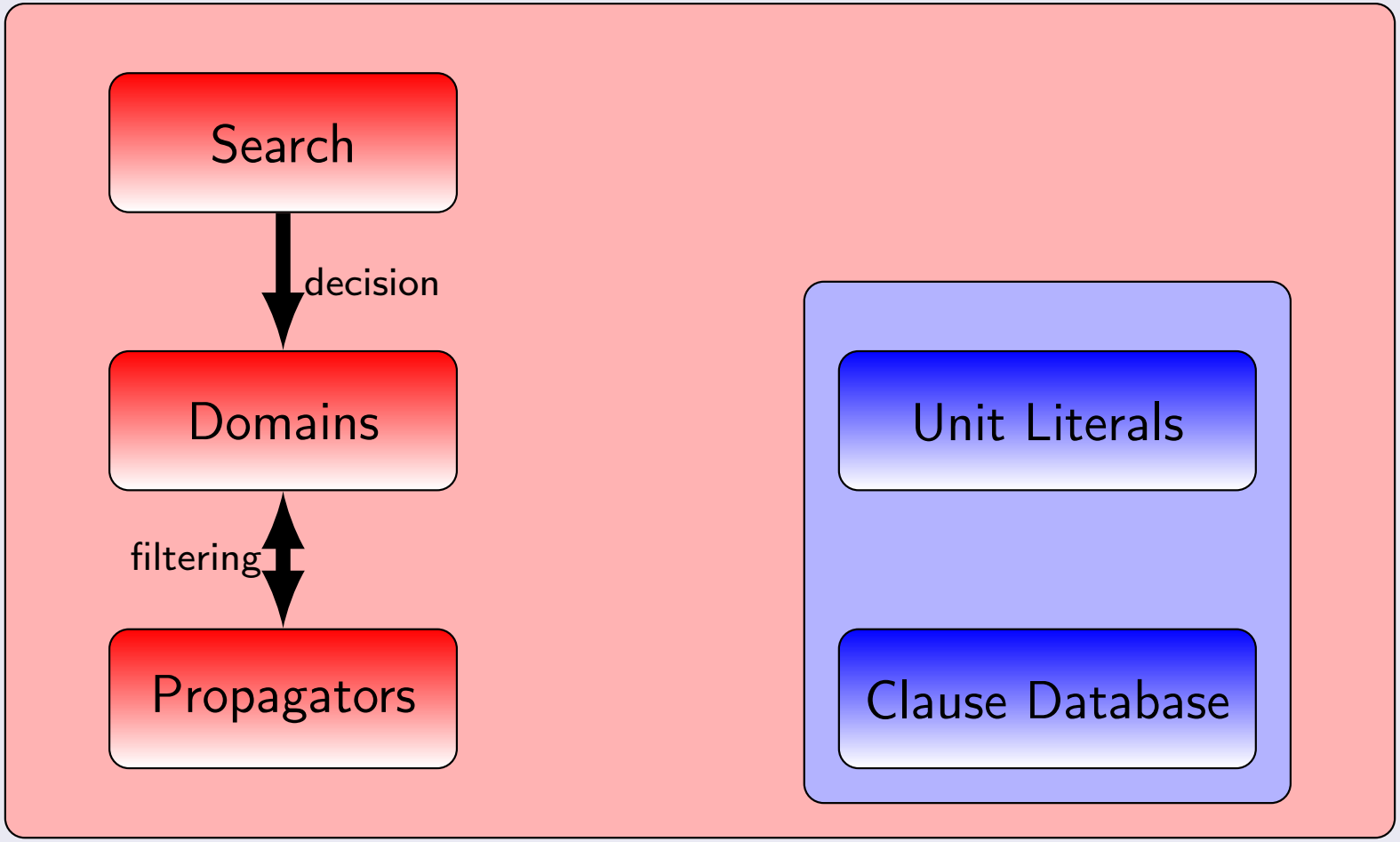
La génération de clauses retardée

Architecture



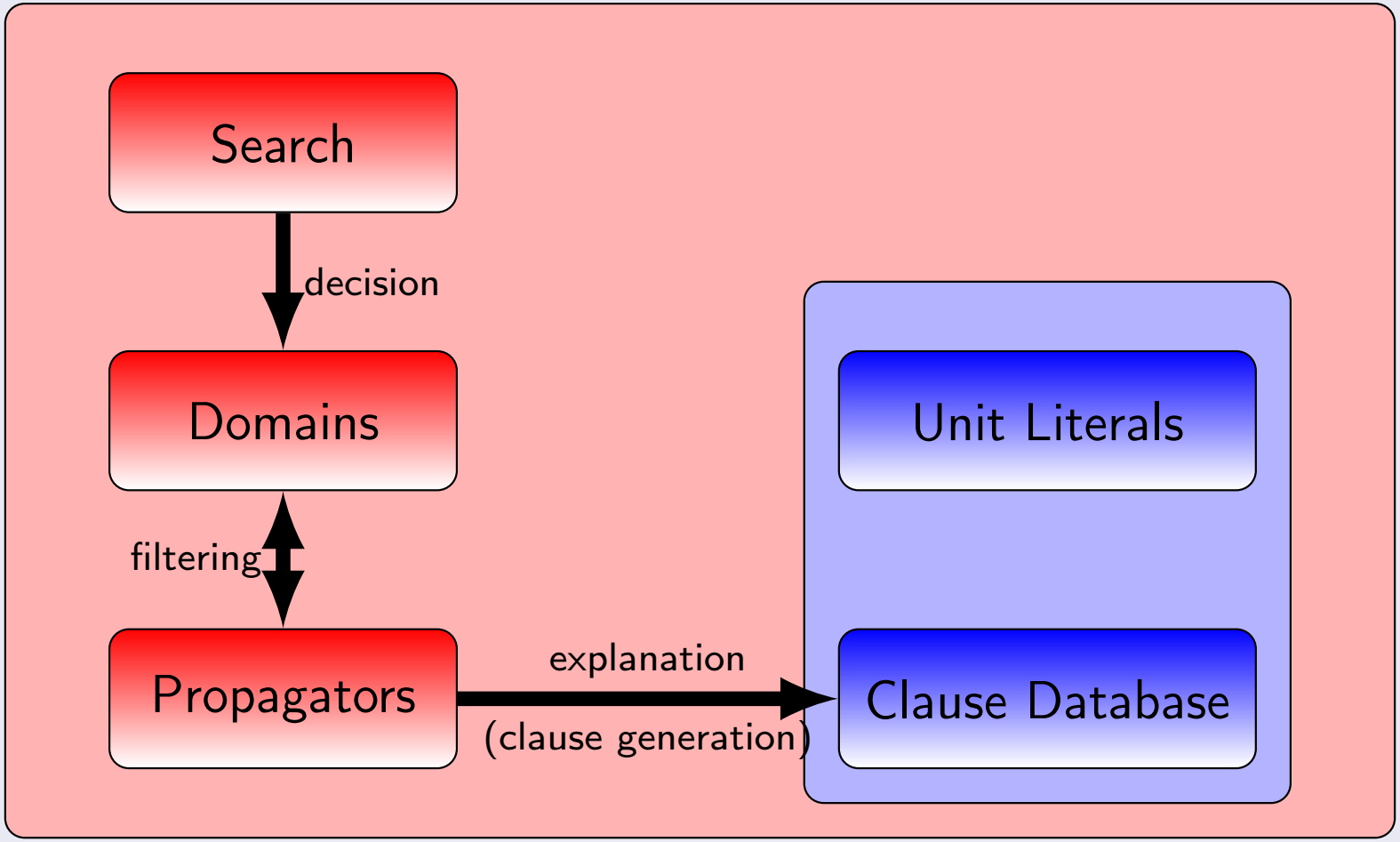
La génération de clauses retardée

Architecture



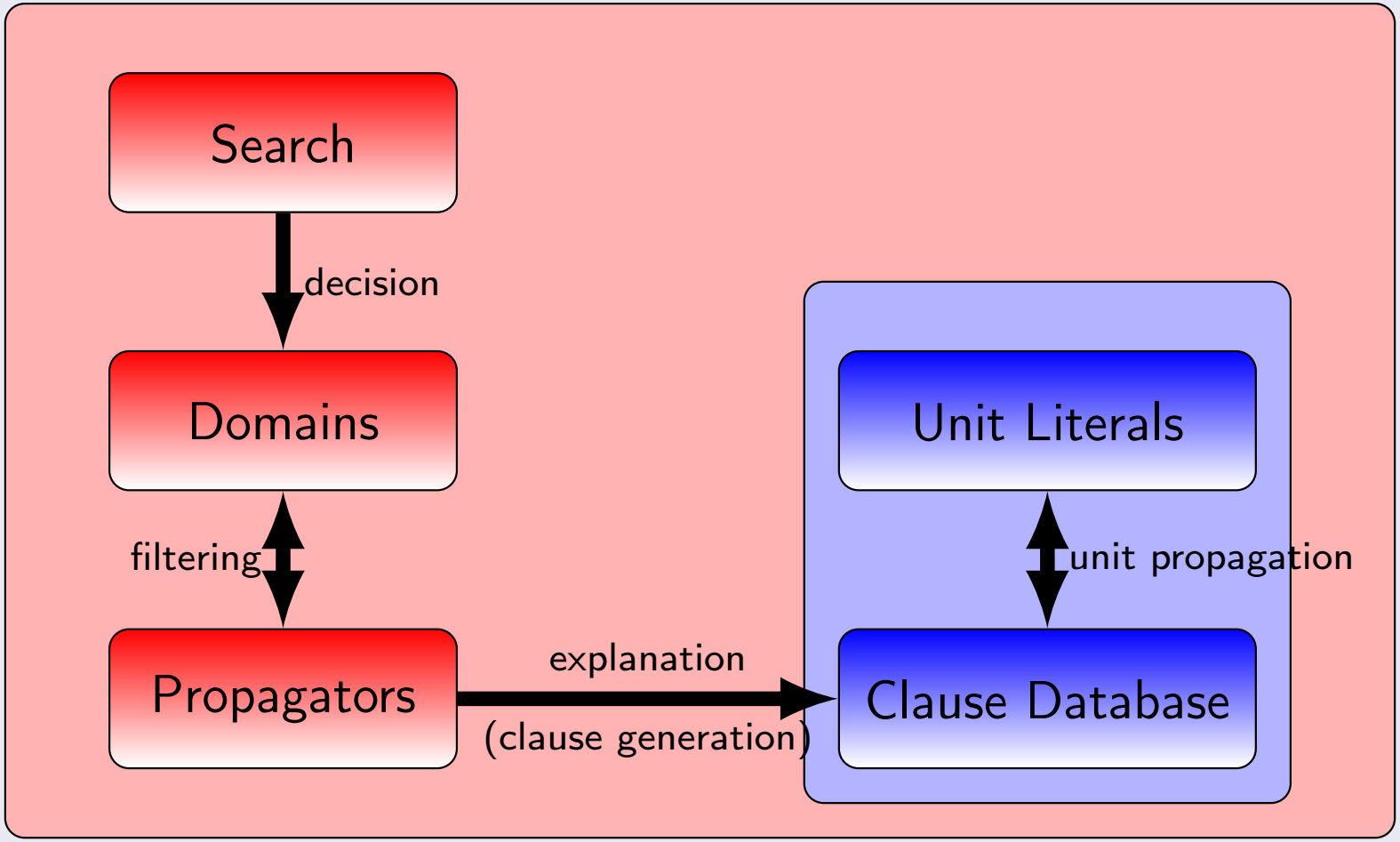
La génération de clauses retardée

Architecture



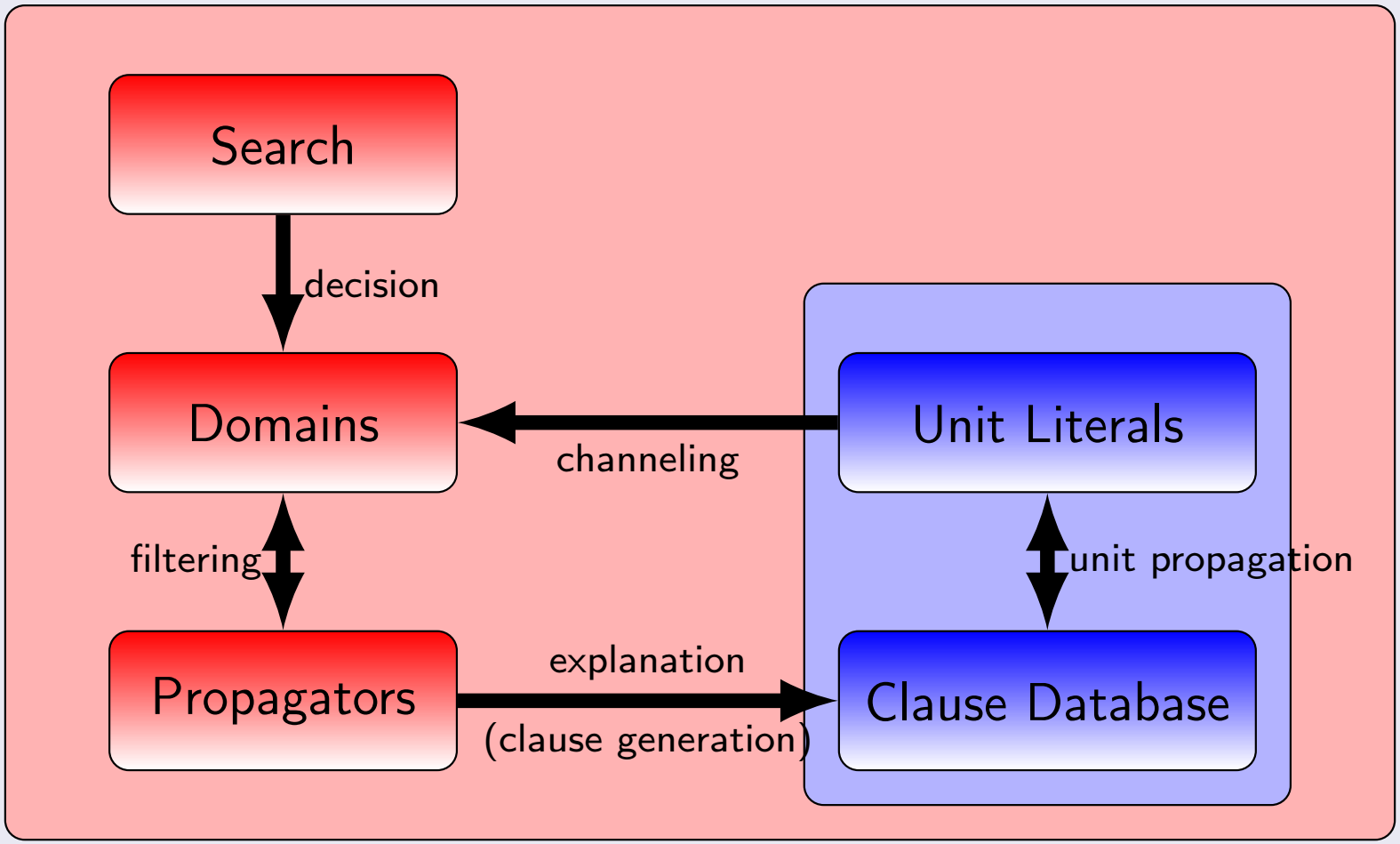
La génération de clauses retardée

Architecture



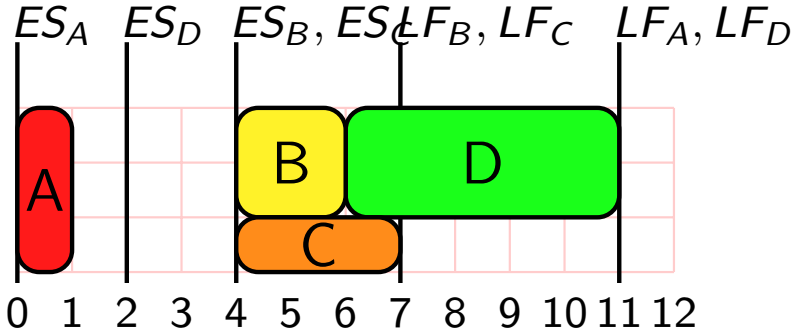
La génération de clauses retardée

Architecture



Application à l'ordonnancement

Génération retardée de clauses [Schutt et al. 2009,2011,2012] basée sur le concept de no-good généralisé [Katsirelos 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 2015], Failure Directed Search [Vilím 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

Synthèse

- PLNE : les formulations indexées par le temps ont les meilleures relaxations avec FS$\succ$DDT$\succ$DT
- Formulations compactes : peuvent être les seules alternatives PLNE pour les grands horizons
  - flots : instances disjonctives
  - événements : instances cumulatives
  - inégalités valides absolument nécessaires
- PLNE et CP/LCG
  - CP/LCG très dominant sur la résolution exacte des instances disjonctives
  - Complémentarité CP/LCG/PLNE pour le calcul de bornes inférieures.
  - Complémentarité CP/LCG/PLNE pour résolution exacte des instances cumulatives