Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes

Arnaud Malapert

Le 9 septembre 2011

à l'École Nationale Supérieure des Techniques Industrielles et des Mines de Nantes

Directeur de thèse :
Narendra Jussien, Professeur, École des Mines de Nantes
Co-encadrant :
Louis-Martin Rousseau, Professeur, École Polytechnique de Montréal
Responsables scientifiques :
Christelle Guéret, Maître-assistante, École des Mines de Nantes
André Langevin, Professeur, Polytechnique de Montréal

UNIVERSITÉ DE NANTES — ECOLE DES MINES DE NANTES — lina — CIRRELT

Plan

Programmation par contraintes

Problème de satisfaction de contraintes (CSP)

- Un ensemble fini de variables ;
- Une fonction associe à chaque variable son domaine fini, l'ensemble discret de valeurs auxquelles elle peut être instanciée ;
- Un ensemble fini de contraintes.
  - Une contrainte est une relation logique établie entre différentes variables.
  - Elle restreint les valeurs que ses variables peuvent prendre simultanément.

Terminologie

- Une **instanciation** assigne une valeur de son domaine à une variable.
- Une **affectation** est l'ensemble des domaines courants de toutes les variables.
- Une **affectation** est dite **consistante** si elle ne viole aucune contrainte.
- Une **solution** est une affectation totale et consistante.

Optimisation sous contraintes : COP = CSP + fonction objectif f

f est souvent modélisée par une variable.

Problèmes d'ordonnancement

Définition

Organiser un ensemble de tâches soumises à certaines contraintes, et dont l'exécution nécessite des ressources :

- déterminer leurs dates de démarrage et d'achèvement ;
- leur attribuer des ressources ;
- de telle sorte que les contraintes soient respectées.

Des problèmes très variés

- Tâches interruptibles, durées fixes ...
- Ressources renouvelables, consommables ...
- Contraintes temporelles, fenêtres de temps ...
- Critères d'optimalité liés au temps, aux ressources, à d'autres coûts ...

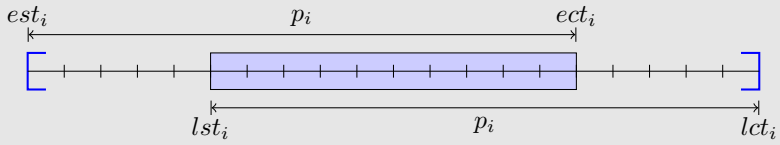
Problèmes traités

Optimisation de critères réguliers pour des problèmes d'atelier et de fournées.

Tâche ou activité

Définitions

- Une tâche T_i est une entité élémentaire de travail localisée dans le temps par une date de début s_i et une date de fin e_i.
- Sa réalisation est caractérisée par une durée positive p_i.
- Une tâche T_i peut être :
  - interruptible $\Rightarrow s_i + p_i \leq e_i$;
  - non interruptible $\Rightarrow s_i + p_i = e_i$.
- Fenêtre de temps d'une tâche T_i : $[est_i, lct_i]$.
- Partie obligatoire d'une tâche non interruptible T_i : $[lst_i, ect_i]$.



Contraintes temporelles

Contrainte de précédence

Une contrainte de précédence entre deux tâches T_i et T_j est représentable par une unique inégalité de potentiel.

Définition : $T_i \preceq T_j \quad \Leftrightarrow \quad s_j - e_i \geq 0 \qquad (T_i \text{ précède } T_j)$.

Contrainte de disjonction

Une contrainte de disjonction entre deux tâches T_i et T_j est satisfaite si les tâches s'exécutent dans des fenêtres de temps disjointes.

Définition : $T_i \simeq T_j \quad \Leftrightarrow \quad (T_i \preceq T_j) \vee (T_j \preceq T_i)$.

Arbitrage : déterminer l'ordre relatif entre les deux tâches.

Réification : une variable booléenne représente l'arbitrage.

Contraintes de partage de ressource

Définitions

1. Une **ressource** est un moyen technique ou humain requis pour la réalisation d'une tâche et disponible en quantité limitée.
2. Une **ressource disjonctive** ne peut exécuter qu'une seule tâche à la fois.

Contrainte globale

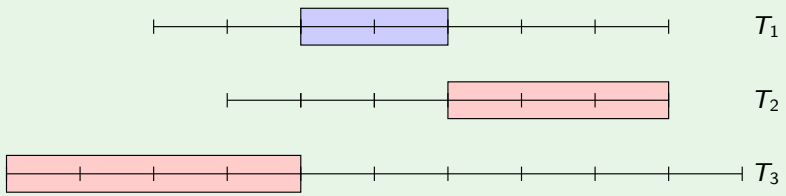
- Utiliser l'information sémantique issue de raisonnements sur des sous-pbs.
- Augmenter l'efficacité du filtrage ou réduire les temps de calcul.

Résolution d'un CSP

Déclaration d'un CSP

$$\texttt{disjunctive}(T_1, T_2, T_3)$$
$$T_1 \in [2, 9] \quad T_2 \in [3, 9] \quad T_3 \in [0, 10]$$

Processus de résolution



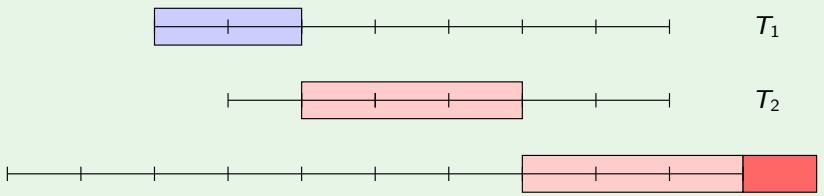
Résolution d'un CSP

Not first/not last

Déterminer si une tâche T_i peut être ordonnancée avant/après un sous-ensemble de tâches Ω :

1. estimer la date de fin au plus tôt ou de début au plus tard des tâches de Ω ;
2. ajuster la fenêtre de temps de la tâche T_i en fonction.

Processus de résolution



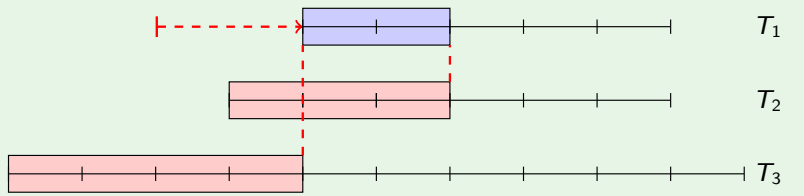
Résolution d'un CSP

Not first/not last

Déterminer si une tâche T_i peut être ordonnancée avant/après un sous-ensemble de tâches Ω :

1. estimer la date de fin au plus tôt ou de début au plus tard des tâches de Ω ;
2. ajuster la fenêtre de temps de la tâche T_i en fonction.

Processus de résolution



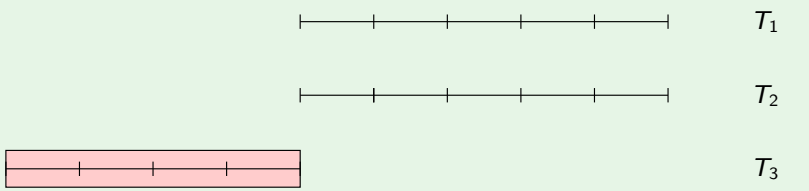
Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes 9 / 51

Résolution d'un CSP

Propagation de contraintes

- La consistance locale ne peut plus réaliser aucune inférence.
- Le point fixe est atteint.

Processus de résolution



Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes 9 / 51

Résolution d'un CSP

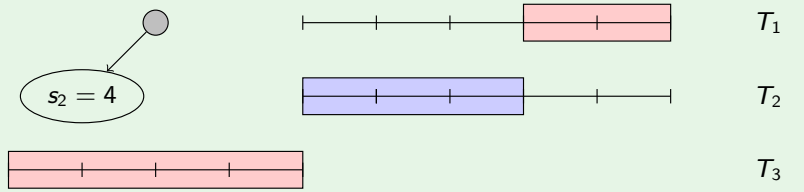
Algorithme backtrack avec standard labelling

Sélection de variable : déterminer l'ordre suivant lequel on va les instancier.

Sélection de valeur : déterminer la prochaine valeur du domaine à tester.

Backtrack : Remettre en cause la dernière décision.

Processus de résolution



Techniques d'ordonnancement d'atelier et de fournées basées sur la programmation par contraintes 9 / 51

Résolution d'un CSP

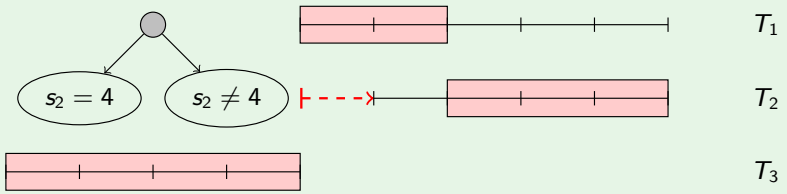
Algorithme backtrack avec standard labelling

Sélection de variable : déterminer l'ordre suivant lequel on va les instancier.

Sélection de valeur : déterminer la prochaine valeur du domaine à tester.

Backtrack : Remettre en cause la dernière décision.

Processus de résolution



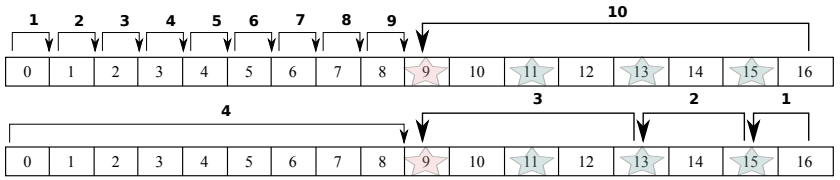
Résolution d'un COP

Procédures d'optimisation : résolution d'une série de CSPs.

bottom-up incrémente la borne inférieure *lb* lorsque le CSP où $obj = lb$ est insatisfiable jusqu'à ce qu'une solution optimale soit trouvée.

top-down résout le CSP où $obj < ub$ jusqu'à ce que le sous-problème devienne insatisfiable.

variantes dichotomiques, incomplètes ...



Définition des problèmes de base

Structure d'un atelier

- Une pièce doit être usinée ou assemblée sur différentes machines.
- Les tâches sont regroupées en n lots constitués chacun de m tâches à exécuter sur m machines distinctes.
- Chaque machine ne peut exécuter qu'une tâche à la fois.

Séquencement des lots

flow-shop : l'ordre est fixé et commun à tous les lots.

job-shop : l'ordre est fixé, mais propre à chaque lot.

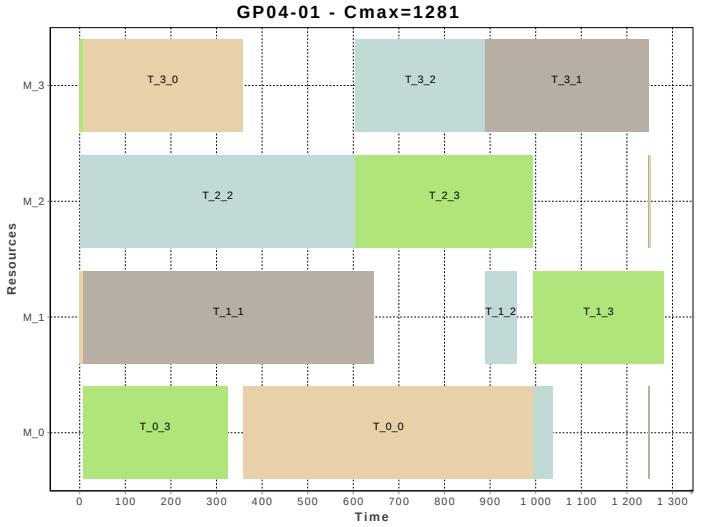
open-shop : le séquencement des tâches des lots n'est pas imposé.

Minimisation du délai total (C_{max}).

$O//C_{max}$, $J//C_{max}$ et $F//C_{max}$ sont NP-difficiles pour $m \geq 3$.

- Sous-ensembles dominants pour d'autres critères réguliers.
- Calcul des chemins critiques et des goulots d'étranglement.

Ordonnancement optimal d'un open-shop



Modèle disjonctif [Roy and Sussman, 1964]

graphe disjonctif $\mathcal{G} = (\mathcal{T}, \mathcal{P}, \mathcal{D})$

- $\mathcal{T}$ est l'ensemble des **noeuds**, représentant les tâches, T_{start} et T_{end}.
- $\mathcal{P}$ est l'ensemble des **arcs** représentant les contraintes de précédence.
- $\mathcal{D}$ est l'ensemble des **arêtes** représentant les disjonctions.

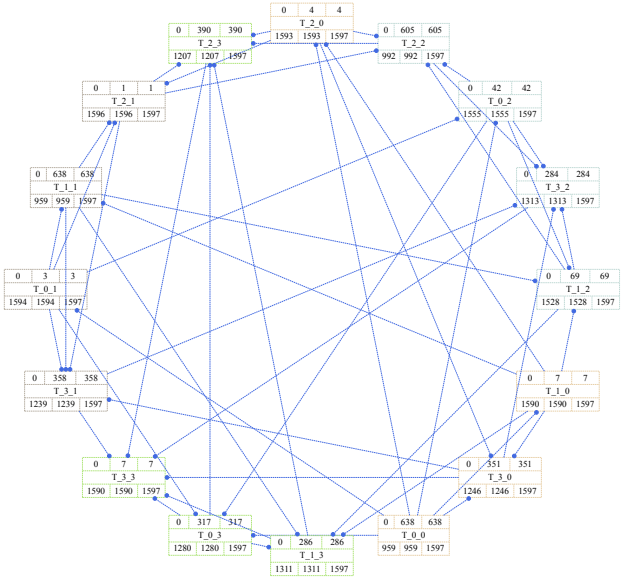
Arbitrage d'une disjonction

Transformation d'une arête $T_i \simeq T_j$ en un arc $T_i \preceq T_j$ ou $T_j \preceq T_i$.

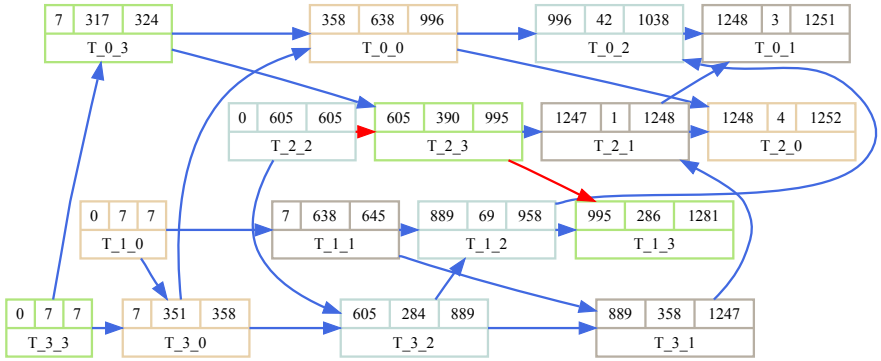
Propriétés

1. L'existence d'un cycle entraîne une inconsistance triviale du problème.
2. Tout ordonnancement réalisable correspond à un arbitrage complet de $\mathcal{G}$.
3. En l'absence d'autres contraintes, il existe un ordonnancement réalisable dont le délai total correspond à la longueur d'un plus long chemin de $\mathcal{G}$, appelé chemin critique.
4. Ici, le calcul des chemins critiques peut être réalisé en temps polynomial.

Construction d'un graphe disjonctif



Arbitrage complet d'un graphe disjonctif



État de l'art sur le problème d'open-shop

Méthodes exactes

- [Brucker et al., 1997] : rech. arborescente complexe (arbitrages multiples).
- [Guéret et al., 2000] : rech. de Brucker avec retour arrière intelligent.
- [Dorndorf et al., 2001] : rech. de Brucker avec tests de consistance.
- [Laborie, 2005] : rech. arborescente pour l'ordonnancement cumulatif.
- [Tamura et al., 2006] : encodage du COP en un problème SAT.

Méthodes approchées

- [Prins, 2000] : algorithme génétique.
- [Blum, 2005] : optimisation par colonies de fourmis.
- [Sha and Hsu, 2008] : optimisation par essaim de particules.

Motivations

- L'absence de séquencement des lots augmente la combinatoire (open-shop).
- Proposer une approche top-down simple et compétitive.

Algorithme de résolution

Composantes clés

1. Calcul d'une solution initiale.
   - La qualité de la première solution découverte a une importance cruciale.
2. Modèles `Light` et `Heavy`.
3. Redémarrages avec enregistrement de *nogoods*.

Un algorithme simple

- Nombre restreint de composantes.
- Composantes disponibles dans plusieurs solveurs.

Modèles `Light` et `Heavy`

Modèle `Light`

- Contraintes de précédence.
- Contraintes de disjonction « réifiées ».
- Stratégie de branchement `slack/wdeg` : arbitrage déterministe d'une disjonction choisie par apprentissage.

Modèle `Heavy`

- Contraintes de précédence.
- Contraintes de disjonction « réifiées ».
- Contraintes globales disjonctives sur les jobs et les machines.
- Stratégie de branchement `profile` : arbitrage randomisée d'une disjonction entre deux tâches participant majoritairement à un pic de demande probabiliste (ressource – instant).

Nouvelle heuristique d'arbitrage : `slack/wdeg`

`slack`

Sélectionner une variable $b_{ij \preceq kl}$ dont la somme des tailles des fenêtres de temps est minimale :

$$(lst_{ij} - est_{ij} + 1) + (lst_{kl} - est_{kl} + 1)$$

`wdeg`

Sélectionner la variable dont le poids pondéré des contraintes est maximal.

- Le poids d'une contrainte est initialisé à la valeur de son degré.
- Il est incrémenté à chaque fois qu'elle déclenche une contradiction.
- Le degré des variables $b_{ij \preceq kl}$ est unitaire.

`slack/wdeg`

Sélectionner une variable $b_{ij \preceq kl}$ dont le quotient du « domaine » sur le degré pondéré est minimal.

Techniques de redémarrages

Politiques de redémarrage

Luby(1,2) : $1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, \ldots$ [Luby et al., 1993]

Walsh(1, 2) : $1, 2, 4, 8, 16, 32, 64, \ldots$ [Walsh, 1999]

Enregistrement de nogoods [Lecoutre et al., 2007]

- L'exploration accomplie durant une étape est totalement mémorisée.
- Cette partie de l'arbre de recherche ne sera plus visitée lors des étapes suivantes.

Diversification de la recherche

Requiert un élément de randomisation ou d'apprentissage.

Expérimentations sur l'open-shop

Jeux d'instances ($\geq 6 \times 6$)

- **Taillard** : 40 instances entre 7×7 et 20×20.
  - Preuve d'optimalité : $OPT = LB$.
  - Les métaheuristiques ferment le jeu d'instances.
- **Brucker et al.** : 26 instances entre 6×6 et 8×8.
- **Guéret-Prins** : 40 instances entre 6×6 et 10×10.

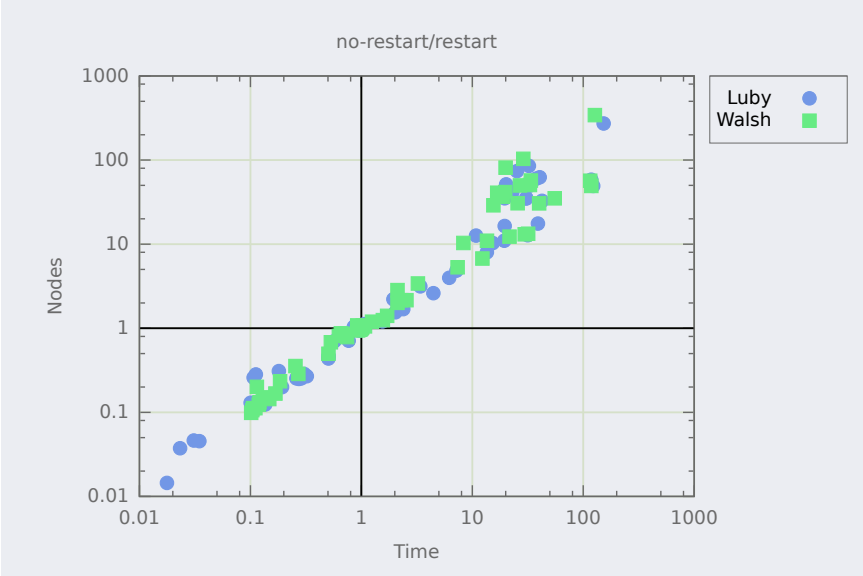
Motivations

- Évaluation des différentes composantes de l'algorithme.
- Comparaison des modèles `Light` et `Heavy`.
- Comparaison avec les approches de l'état de l'art.

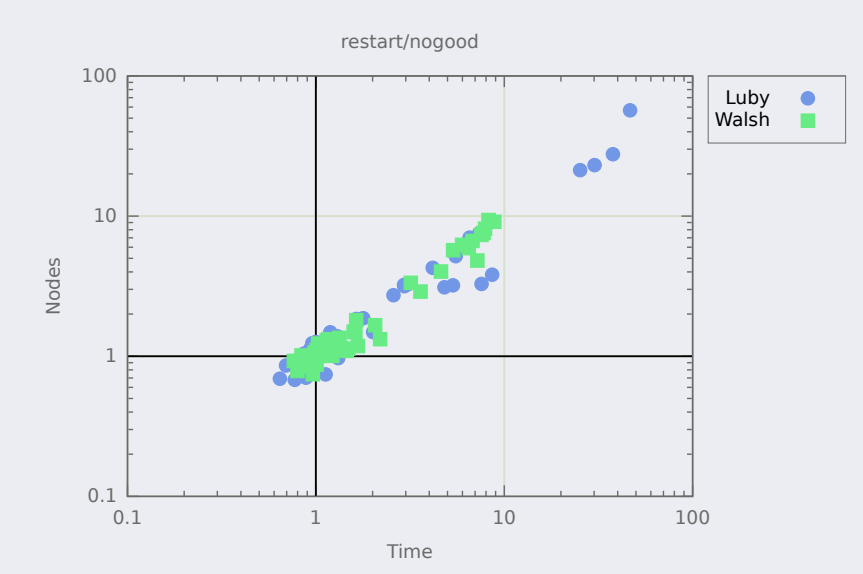
Protocole

- 20 exécutions de l'algorithme.
- Solveur PPC : `choco` et `mistral`.
- Deux grilles de calcul avec des machines Linux.

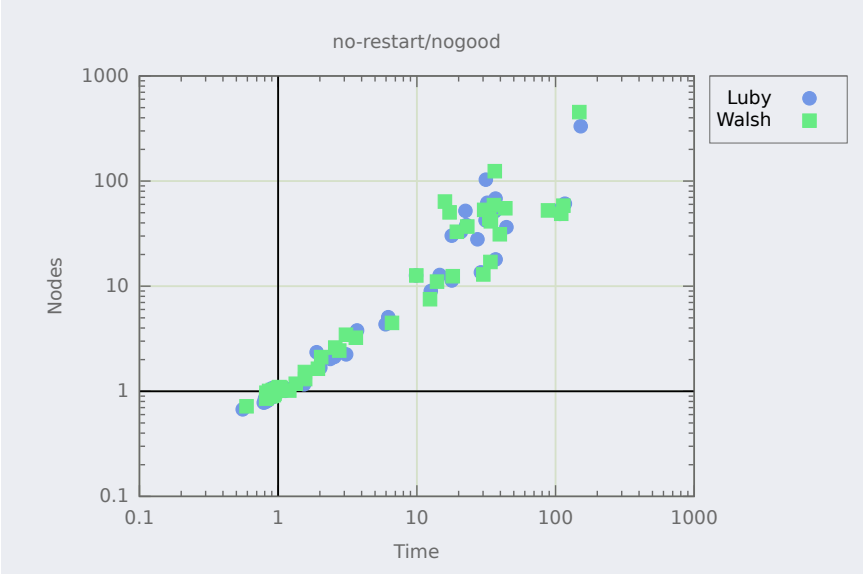
Redémarrages



Enregistrement des nogoods



Redémarrages avec enregistrement des nogoods



Comparaison des modèles `Light` et `Heavy`

Modèles robustes

Ils résolvent toutes les instances pour toutes les exécutions.

Différences pendant la résolution

- Le modèle `Light` est plus rapide.
- Le modèle `Heavy` visite moins de nœuds.

Complémentarité des contraintes et du branchement

- `Light` + `profile` : la qualité de la courbe de demande probabiliste est dégradée, car les fenêtres de temps sont plus larges.
- `Heavy` + `slack/wdeg` : les contradictions relevées par les contraintes globales perturbent l'apprentissage des poids des disjonctions.

	Light		Heavy		Light-profile		Heavy-slack/wdeg	
	$\bar{t}$	$\bar{n}$	$\bar{t}$	$\bar{n}$	$\bar{t}$	$\bar{n}$	$\bar{t}$	$\bar{n}$
hard11	**286.1s**	1.3M	428.9s	**0.4M**	1774.3	2.5M	806.6	0.6M

Comparaison avec les approches de l'état de l'art

Métaheuristiques

- Certains optimums ne sont découverts par aucune métaheuristique.

Name		Heavy	ACO	PSO
`tai20-*`	$\bar{t}$	103.6s	52.0s	**35.1s**
`gp10-*`	$\bar{t}$	**13.1s**	567.1s	367.3s

Méthodes exactes

- Par rapport aux nôtres, les résultats de [Laborie, 2005] semblent :
  - Instances de Taillard : pas de résultats ;
  - Instances de Brucker : comparables ;
  - Instances de Guéret-Prins : légèrement moins bons.
- [Tamura et al., 2006]

hard11	Light	Heavy	SAT
$\bar{t}$	286.1s	428.9s	$\bar{t} \gg$ **2843.5s**

Conclusion sur les problèmes d'atelier

Contributions

- Algorithme simple, flexible, robuste et générique.
  - Identification des composantes clés de l'algorithme.
  - Modèles `Light` et `Heavy`.
  - Problèmes d'open-shop et de job-shop.
- Nouvelles heuristiques d'arbitrage basées sur les domaines et degrés.
- Réflexion sur l'implémentation du modèle disjonctif dans un solveur

Définition du problème de base

Tâches

Chaque tâche j est caractérisée par sa durée d'exécution p_j, sa date échue d_j, et sa taille s_j (les tailles des tâches ne sont pas identiques).

Machine à traitement par fournées

Elle peut exécuter simultanément plusieurs tâches dans une fournée :

- aucune tâche n'est ajoutée ou retirée d'une fournée pendant son exécution ;
- la somme des tailles des jobs d'une fournée ne dépasse pas sa capacité b ;
- la durée d'exécution d'une fournée est égale à celle de sa plus longue tâche ;
- la date de fin C_j d'une tâche j est la date de fin de sa fournée.

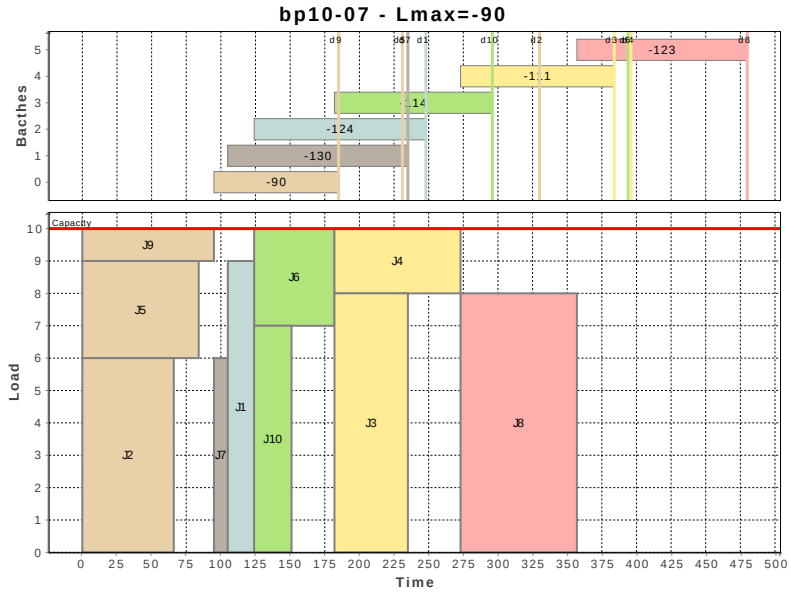
Hypothèse

Les tâches et la machine sont disponibles depuis l'instant 0.

Minimiser le retard algébrique maximal $L_{max} = \max_{1 \leq j \leq n}(C_j - d_j)$

$1|p\text{-}batch; b < n; non\text{-}identical|L_{max}$ est NP-difficile.

Ordonnancement optimal de fournées



État de l'art

Problèmes de fournées [Potts and Kovalyov, 2000]

- Premiers travaux : complexité et algorithmes d'approximation.
- Critères d'optimalité dépendant des dates d'achèvement (C_{max}, $\sum C_j$, $\sum w_j C_j$) : les tâches ont quelquefois des tailles différentes.
- Critères d'optimalité dépendant des dates échues : les tâches ont généralement des tailles identiques ($s_j = 1$).
- [Vilím, 2007] : algorithmes de filtrage pour un problème de fournées *s-batch* avec des familles de tâches et des temps d'attente dépendant de la séquence.

$1|p\text{-}batch; b < n; non\text{-}identical|L_{max}$

- PLNE basée sur le placement et le séquencement [Daste et al., 2008b].
- Branch-and-price (problème maître de séquencement) [Daste et al., 2008a].

Motivations

- Proposer un modèle PPC exploitant la décomposition naturelle du problème.
- Appliquer des techniques de filtrage basé sur les coûts.

Principe de la décomposition

Placement des jobs dans les fournées

Le problème de placement (*bin packing*) à une dimension est NP-Complet : soit n articles (tâches) caractérisés chacun par une taille positive $s_j \geq 0$ et m conteneurs (fournées) caractérisés par une capacité b, existe-t'il un placement des n articles dans les m conteneurs tel que la somme des tailles des articles dans chaque conteneur reste inférieure à la capacité b ?

Minimisation du retard algébrique maximal des fournées sur une machine

Le problème $1||L_{max}$, peut être résolu en temps polynomial :
un ordonnancement optimal est obtenu en appliquant la règle de Jackson (*earliest due date (EDD-)rule*) qui séquence les tâches en ordre non décroissant de leur date échue.

Modélisation des fournées

Constraintes

- Contraintes ensemblistes sur les durées et les dates échues des fournées.
- Contrainte globale de placement à une dimension : `pack` [Shaw, 2004].
- Contrainte globale d'optimisation de la séquence : `sequenceEDD`.
- Contraintes basiques d'élimination de symétries sur le placement des tâches.

Algorithme de résolution

Structure d'une solution

- Un placement réalisable des tâches dans les fournées.
- Le retard algébrique maximal des fournées sur une machine.

Algorithme de recherche

- Procédure d'optimisation : top-down.
- Algorithme backtrack sans redémarrages.
- Méthode de séparation : standard labelling n-aire.
  - Sélection de variables : complete decreasing.
  - Sélection de valeurs : first fit, best fit, batch fit.
- Élimination dynamique de symétries sur le placement des tâches.

Description de la contrainte globale `sequenceEDD`

Définition

la contrainte globale `sequenceEDD` assure que la valeur de la variable L_{max} soit le plus grand retard algébrique de la séquence de fournées ordonnancées en suivant la règle EDD.

Raisonnements liés à la satisfiabilité

Aucun, toutes les séquences de fournées sont réalisables dès lors que le placement des tâches est consistant.

Raisonnements liés à l'optimalité

- L'idée principale est de déduire des contraintes primitives à partir des informations sur les coûts des affectations.
- `sequenceEDD` utilise une relaxation vers un problème à une machine.

Règles de filtrage

Relaxation

- Une instance $I(\mathcal{A})$ du problème $1||L_{max}$ est construite à partir d'une affectation $\mathcal{A}$, puis ordonnancée en suivant la règle EDD.
- Le retard de la relaxation ne décroît jamais après une décision et converge vers la valeur de l'objectif au cours de la recherche.

Règles de filtrage simples ($O(n)$)

FF : appliquer la relaxation après le placement des tâches.

LF : appliquer la relaxation à chaque nœud de l'arbre de recherche.

Règles de filtrage basé sur les coûts ($O(n^2)$)

AF : Restrictions sur le placement des tâches.

PF : Restrictions sur le nombre de fournées non vides.

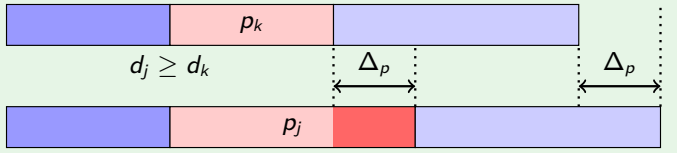
1. Augmenter le retard en fonction des nouvelles fournées nécessaires.
2. Réduire le nombre de nouvelles fournées pour satisfaire les coûts.

Placement d'une tâche (AF)

Règle de filtrage basé sur les coûts

1. Calculer le coût marginal du placement de la tâche j dans la fournée k.
2. Additionner le coût marginal et le retard algébrique de la relaxation.
3. En cas de dépassement du coût, interdire le placement.

Calcul du coût marginal ($p_j \geq p_k$)



Contribution algorithmique

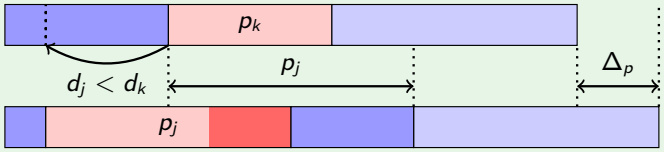
Un algorithme réduit la complexité de $O(n^3)$ à $O(n^2)$.

Placement d'une tâche (AF)

Règle de filtrage basé sur les coûts

1. Calculer le coût marginal du placement de la tâche j dans la fournée k.
2. Additionner le coût marginal et le retard algébrique de la relaxation.
3. En cas de dépassement du coût, interdire le placement.

Calcul du coût marginal ($p_j \geq p_k$)



Contribution algorithmique

Un algorithme réduit la complexité de $O(n^3)$ à $O(n^2)$.

Expérimentations

Jeu d'instances

200 instances générées aléatoirement ($n \in \{10, 20, 50, 75, 100\}$).

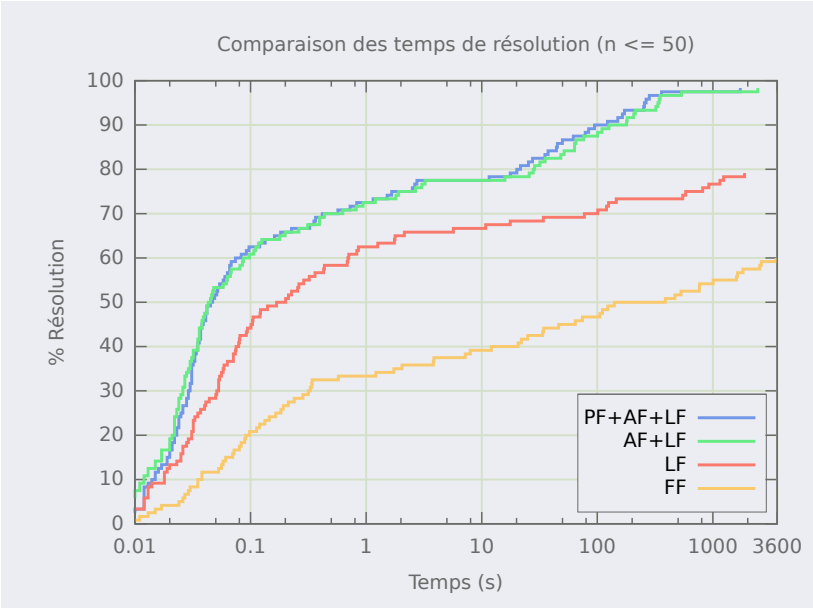
Motivations

- Évaluation des règles de filtrage.
- Évaluation des heuristiques de selection de valeur.
- Comparaison avec :
  - un modèle PLNE [Daste et al., 2008b] ;
  - un branch-and-price [Daste et al., 2008a].

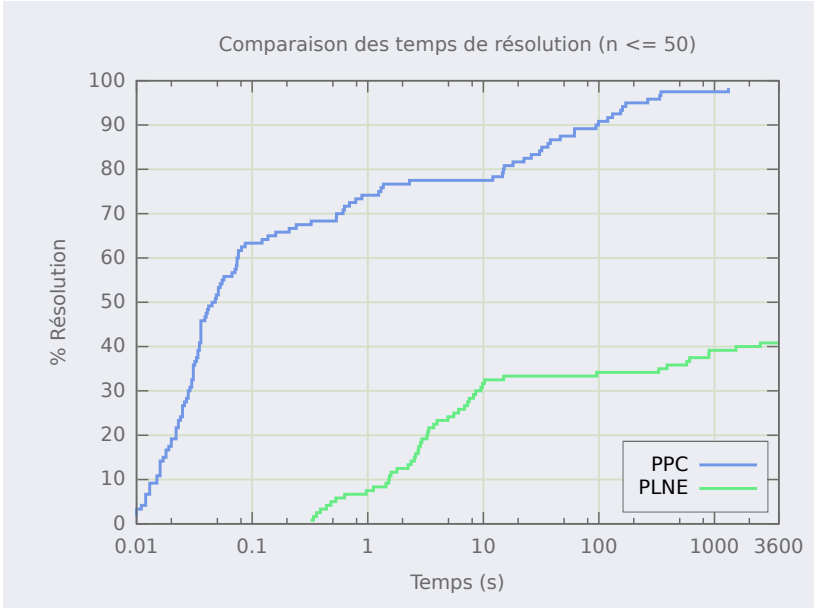
Protocole

- Solveur PPC : choco.
- Solveur PLNE : Ilog Cplex 11.2.1.
- Grille de calcul avec machines Linux (Proc 2.4 GHz, RAM 48 GB).

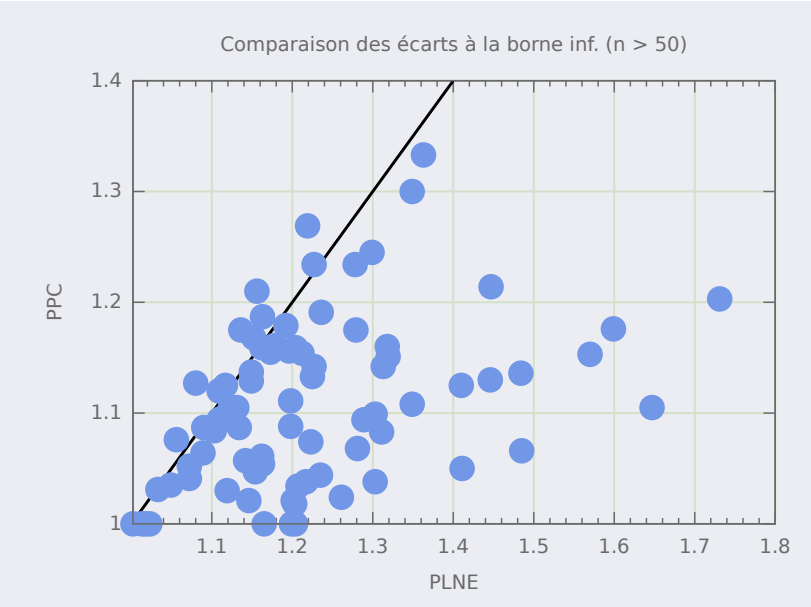
Évaluation des règles de filtrage



Comparaison avec un modèle PLNE



Comparaison avec un modèle PLNE



Comparaison avec un branch-and-price

$n = 20$

B&P 55 % des instances résolues avec une limite de temps d'une heure.

PPC 100 % des instances résolues en moins d'une seconde.

$n = 50$

B&P 8 % des instances résolues avec une limite de temps d'une heure.

PPC 95 % des instances résolues avec une limite de temps d'une heure avec un temps moyen de résolution $\bar{t} \leq 100$s.

Conclusion sur les problèmes de fournées

Contributions

- Exploitation de la décomposition naturelle du problème.
- Contrainte globale pour l'optimisation exploitant une relaxation polynomiale basée sur un problème d'ordonnancement à une machine.
- Stratégie de recherche et heuristique de sélection batch fit.

Solveur `choco`

Un solveur de contraintes sous licence libre

`http://choco.mines-nantes.fr/`

Motivations

Scientifique : diffuser le travail réalisé et susciter des collaborations.

Technique : participer au développement d'un projet open source.

Module d'ordonnancement sous contraintes

TaskVariable

```
makeTaskVar(...); makeTaskVarArray(...);
```

Constraint

```
//Temporelles
precedence(task1, task2, delta);
precedenceDisjoint(task1, task2, dir, fwd, bwd);
//Partage de ressource
disjunctive(tasks[], usages[], uppBound);
cumulative(tasks[],heights[],usages[],consumption,capacity,uppBound)
//Allocation de ressource (en développement)
useResources(task, k, resources[]);
useResourcesGeq(task, k, resources[]);
//Placement
pack(itemSets[], loads[], bins[], sizes[], nbNonEmpty);
```

Prétraitement d'un modèle (reformulation)

Construction automatique et modulaire du modèle disjonctif à partir des fenêtres de temps, des contraintes temporelles et de partage de ressource.

Éléments du solveur

Stratégies de recherche (BranchingFactory)

- Ordonnancement : `profile`, `setTimes`.
- Domaines sur degrés : (`dom` | `slack` | `preserved`)/(`deg` | `ddeg` | `wdeg`)
  - Standard labelling binaire ou n-aire.
  - Calcul incrémental des poids ou non.
- Placement : complete decreasing (first | best | worst) fit + symétries.

Algorithme de recherche (Solver)

```
//Politiques de redémarrage
setRestart(boolean restart);
setGeometricRestart(int base, double grow, int restartLimit);
setLubyRestart(int base, int grow, int restartLimit);
//Enregistrement de nogoods
setRecordNogoodFromRestart(boolean record);
//Procédures d'optimisation (en développement)
Configuration.BOTTOM_UP;
Configuration.INIT_DESTRUCTIVE_LOWER_BOUND;
Configuration.INIT_SHAVING;
```


Conclusion

Problèmes d'atelier : JOPT09, CP09, JoC11

- Algorithme simple, flexible, robuste et générique.
- Modèles `Light` et `Heavy`.
- Heuristique d'arbitrage basées sur les domaines et degrés : `slack/wdeg`.

Problèmes de fournées : JOPT10, CPAIOR11, EJOR

- Modèle basée sur une décomposition naturelle du problème.
- Contrainte globale pour l'optimisation exploitant une relaxation polynomiale.

Solveur `choco` : CP10

- Module d'ordonnancement et de placement à une dimension.
- Allocation de ressources.

Perspectives

Court terme

- Application à d'autres problèmes d'atelier.
  - [Grimes and Hebrard, 2010, Grimes and Hebrard, 2011].
- Application à d'autres problèmes de fournées.
- Heuristiques de sélection basées sur les coûts.

Moyen terme

- Contraintes de partage de ressource :
  1. Adapter dynamiquement les propagateurs ;
  2. Améliorer l'apprentissage des poids pondérés des variables.
- Relaxations basées sur les problèmes à machines parallèles.

Long terme

- Structure disjonctive.
- Allocation de ressources.
- Contraintes souples.

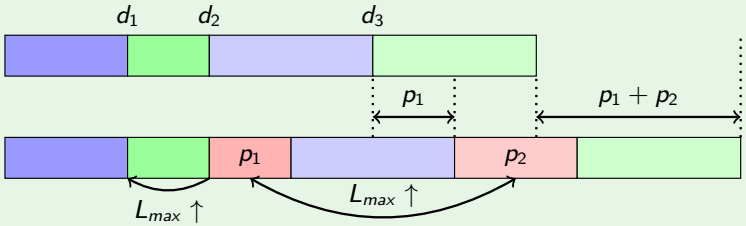
Nombre de fournées (PF)

Règles de filtrage basé sur les coûts

- Mettre à jour le coût associé au nombre minimal de fournées.
- Réduction du nombre maximal de fournées en cas de dépassement des coûts.

Exemple avec 3 tâches et 2 nouvelles fournées

- Insertion de nouvelles fournées consistuées d'une unique tâche.
- Tris : $p_1 \leq p_2 \leq p_3$ et $d_1 \leq d_2 \leq d_3$



References I

Blum, C. (2005).
Beam-ACO : hybridizing ant colony optimization with beam search : an application to open shop scheduling.
Comput. Oper. Res., 32(6) :1565–1591.

Brucker, P., Hurink, J., Jurisch, B., and Wöstmann, B. (1997).
A branch & bound algorithm for the open-shop problem.
In *GO-II Meeting : Proceedings of the second international colloquium on Graphs and optimization*, pages 43–59, Amsterdam, The Netherlands. Elsevier Science Publishers B. V.

Daste, D., Gueret, C., and Lahlou, C. (2008a).
A Branch-and-Price algorithm to minimize the maximum lateness on a batch processing machine.
In *Proceedings of the 11th international workshop on Project Management and Scheduling PMS'08*, pages 64–69, IStanbul Turquie.

Daste, D., Gueret, C., and Lahlou, C. (2008b).
Génération de colonnes pour l'ordonnancement d'une machine à traitement par fournées.
In *7ième conférence internationale de modélisation et simulation MOSIM'08*, volume 3, pages 1783–1790, Paris France.

Dorndorf, U., Pesch, E., and Huy, T. P. (2001).
Solving the open shop scheduling problem.
Journal of Scheduling, 4 :157–174.

Grimes, D. and Hebrard, E. (2010).
Job shop scheduling with setup times and maximal time-lags : A simple constraint programming approach.
In Lodi, A., Milano, M., and Toth, P., editors, *CPAIOR*, volume 6140 of *Lecture Notes in Computer Science*, pages 147–161. Springer.

Grimes, D. and Hebrard, E. (2011).
Models and strategies for variants of the job shop scheduling problem.
In Lee, J. H.-M., editor, *CP*, volume 6876 of *Lecture Notes in Computer Science*, pages 356–372. Springer.

Guéret, C., Jussien, N., and Prins, C. (2000).
Using intelligent backtracking to improve branch-and-bound methods : An application to open-shop problems.
European Journal of Operational Research, 127 :344–354.

References II

Laborie, P. (2005).
Complete MCS-based search : Application to resource constrained project scheduling.
In Kaelbling, L. P. and Saffiotti, A., editors, *IJCAI*, pages 181–186. Professional Book Center.

Lecoutre, C., Sais, L., Tabary, S., and Vidal, V. (2007).
Nogood recording from restarts.
In Veloso, M. M., editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 131–136, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Luby, Sinclair, and Zuckerman (1993).
Optimal speedup of las vegas algorithms.
IPL : Information Processing Letters, 47 :173–180.

Potts, C. N. and Kovalyov, M. Y. (2000).
Scheduling with batching : A review.
European Journal of Operational Research, 120(2) :228–249.

Prins, C. (2000).
Competitive genetic algorithms for the open-shop scheduling problem.
Mathematical methods of operations research, 52(3) :389–411.

Roy, B. and Sussman, B. (1964).
Les problèmes d'ordonnancement avec contraintes disjonctives.
Technical report, Technical Report Note DS no 9bis, SEMA, Paris.

Sha, D. Y. and Hsu, C.-Y. (2008).
A new particle swarm optimization for the open shop scheduling problem.
Comput. Oper. Res., 35(10) :3243–3261.

Shaw, P. (2004).
A constraint for bin packing.
In Wallace, M., editor, *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, volume 3258 of *Lecture Notes in Computer Science*, pages 648–662. Springer.

References III

Tamura, N., Taga, A., Kitagawa, S., and Banbara, M. (2006).
Compiling finite linear CSP into SAT.
In *Principles and Practice of Constraint Programming - CP 2006*, volume 4204 of *Lecture Notes in Computer Science*, pages 590–603. Springer Berlin / Heidelberg.

Vilím, P. (2007).
Global Constraints in Scheduling.
PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, KTIML MFF, Universita Karlova, Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic.

Walsh, T. (1999).
Search in a small world.
In *IJCAI '99 : Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1172–1177, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.