Ordonnancement de tâches dans OAR, de la théorie à la pratique

L. Eyraud, G. Mounié, D. Trystram

ID-IMAG, Grenoble, France

23 Mars 2006

Ordonnancement pour les clusters

Contexte :

- Un (grand) groupe de machines
- Un groupe d'utilisateurs
- Quelques applications parallèles à exécuter

Espérances :

- Optimiser l'utilisation de la machine
- Garantir un temps de réponse rapide
- Garantir une équité entre utilisateurs
- Permettre d'exprimer des contraintes
- ...

Ordonnancement pour les clusters

Une grande problématique pour nous à ID

- Théorique : D. Trystram & co
- Pratique : O. Richard, OAR, Grid'5000

Faible interconnexion entre les deux parties

- Plusieurs algorithmes théoriques
- Aucune implémentation réelle

Pourquoi ?

- Modèles théoriques simplifiés
- Grosse inertie
- Besoin de comprendre les choix d'ordonnancement

Plan de l'exposé

1. Un algorithme théorique
   - Modèle : tâches modelables
   - Un schéma en étagères
2. Le modèle d'ordonnancement dans OAR
   - Du point de vue utilisateur
   - Conséquences
3. Validation

Motivation

Modèle fin :

- Voir chaque application comme un graphe de tâches
- Ordonnancer toutes les tâches en même temps
- Permet d'entrelacer les applications : meilleure utilisation
- Non praticable : trop complexe

Motivation

Modèle fin :

- Voir chaque application comme un graphe de tâches
- Ordonnancer toutes les tâches en même temps
- Permet d'entrelacer les applications : meilleure utilisation
- Non praticable : trop complexe

Modèle plus réaliste (OAR, PBS, ...) :

- Chaque application est une "boîte" de forme fixée
- Ordonnancement = Pavage
- Très rigide, faible expressivité

Motivation

Modèle fin :

- Voir chaque application comme un graphe de tâches
- Ordonnancer toutes les tâches en même temps
- Permet d'entrelacer les applications : meilleure utilisation
- Non praticable : trop complexe

Modèle plus réaliste (OAR, PBS, ...) :

- Chaque application est une "boîte" de forme fixée
- Ordonnancement = Pavage
- Très rigide, faible expressivité

Modèle intermédiaire :

- Forme des "boîtes" non fixées
- Plus de liberté pour l'ordonnanceur

Modèle

- m machines homogènes
- n tâches modelables indépendantes :
  - $p_i(q)$: temps de calcul de la tâche i sur q machines
  - monotonie : pas d'accélération super-linéaire
  - r_i : date d'arrivée (soumission) de la tâche i
  - une priorité ω_i
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Makespan : $C_{\max} = \max_i C_i$
  - Temps de complétion moyen : $\sum_i \omega_i C_i$

Observations

Pour optimiser :

- C_{max} : grosses tâches d'abord, petites dans les trous
- $\sum \omega_i C_i$: petites tâches d'abord ; plus difficile

Observations

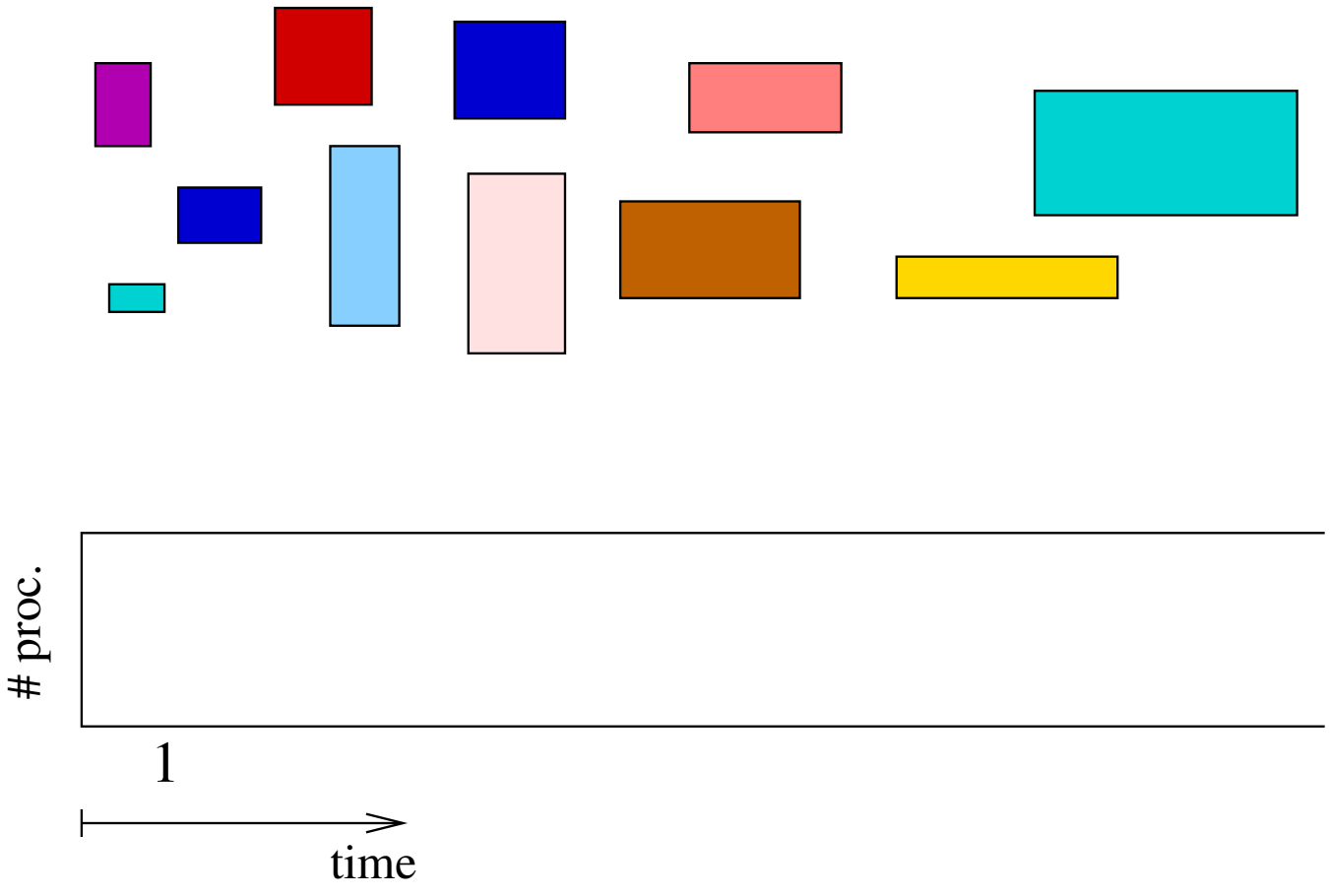
Pour optimiser :

- C_{max} : grosses tâches d'abord, petites dans les trous
- $\sum \omega_i C_i$: petites tâches d'abord ; plus difficile

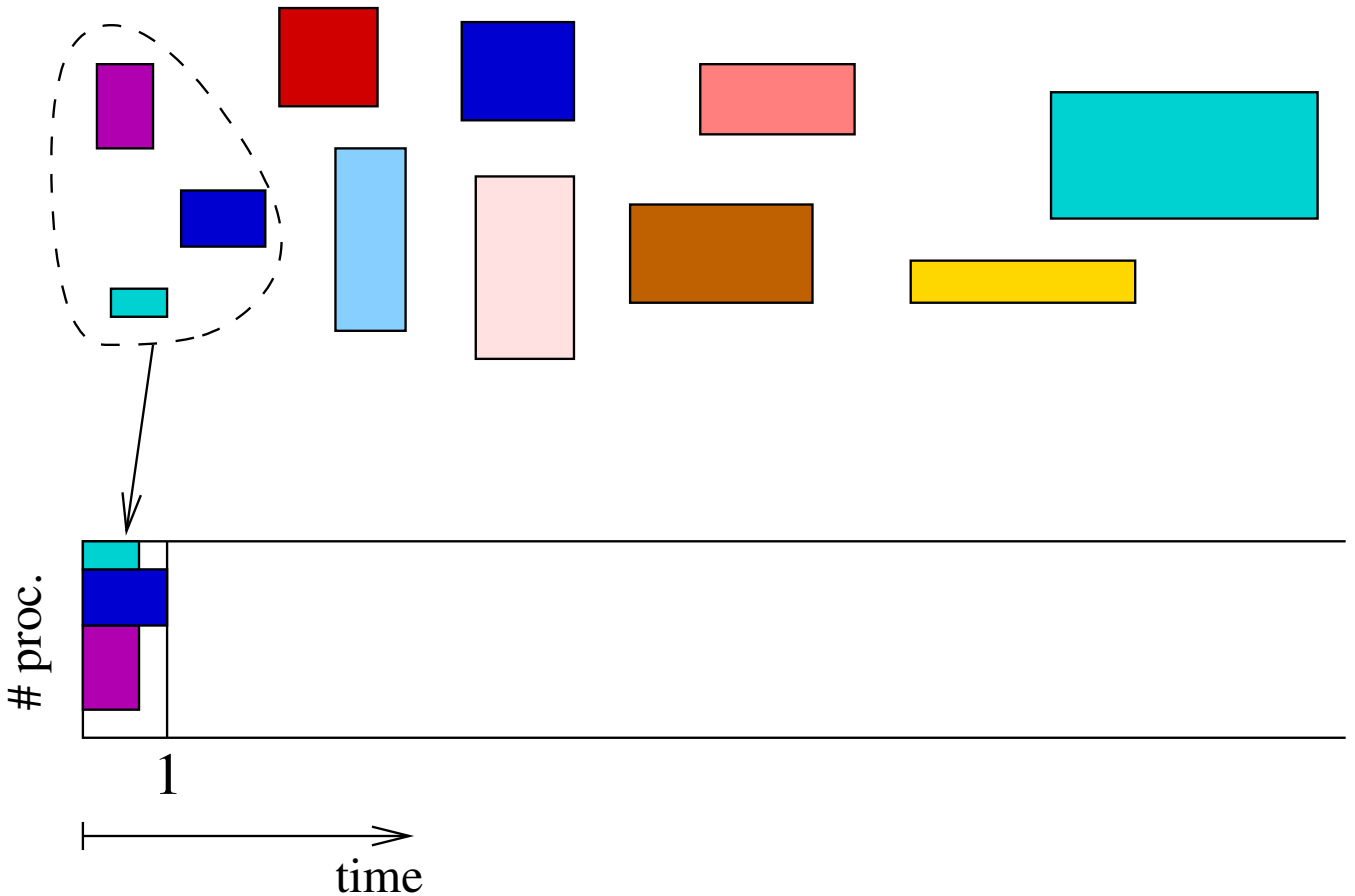
Les deux à la fois :

- Batchs de plus en plus grands
- À l'intérieur, sélectionner le plus de poids possible
- Garantie de performance : $(6, 6)$; $(3, 4.08)$ en moyenne
- Amélioration pratique en compactant l'ordonnancement

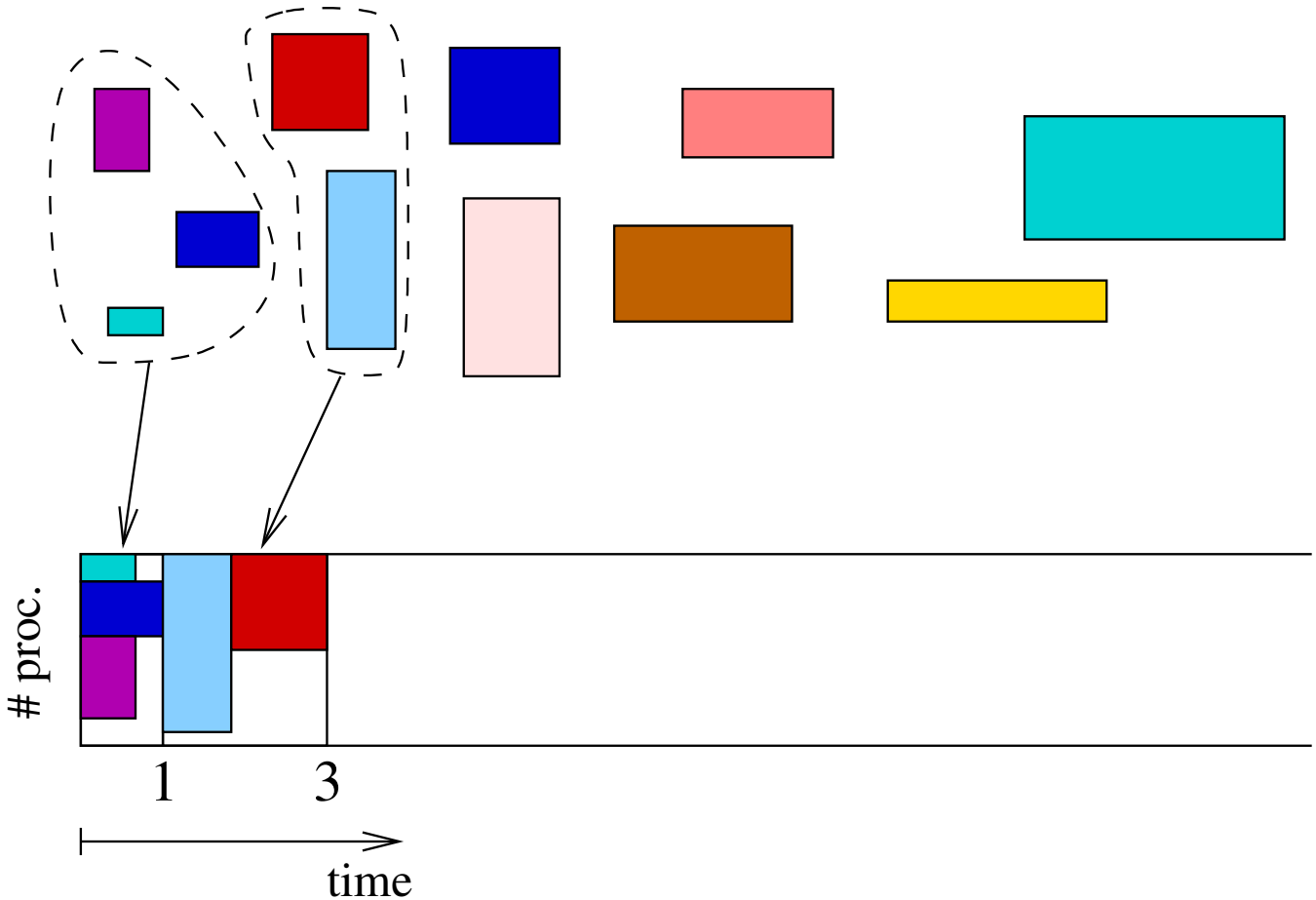
Un dessin vaut mieux que tous les mots



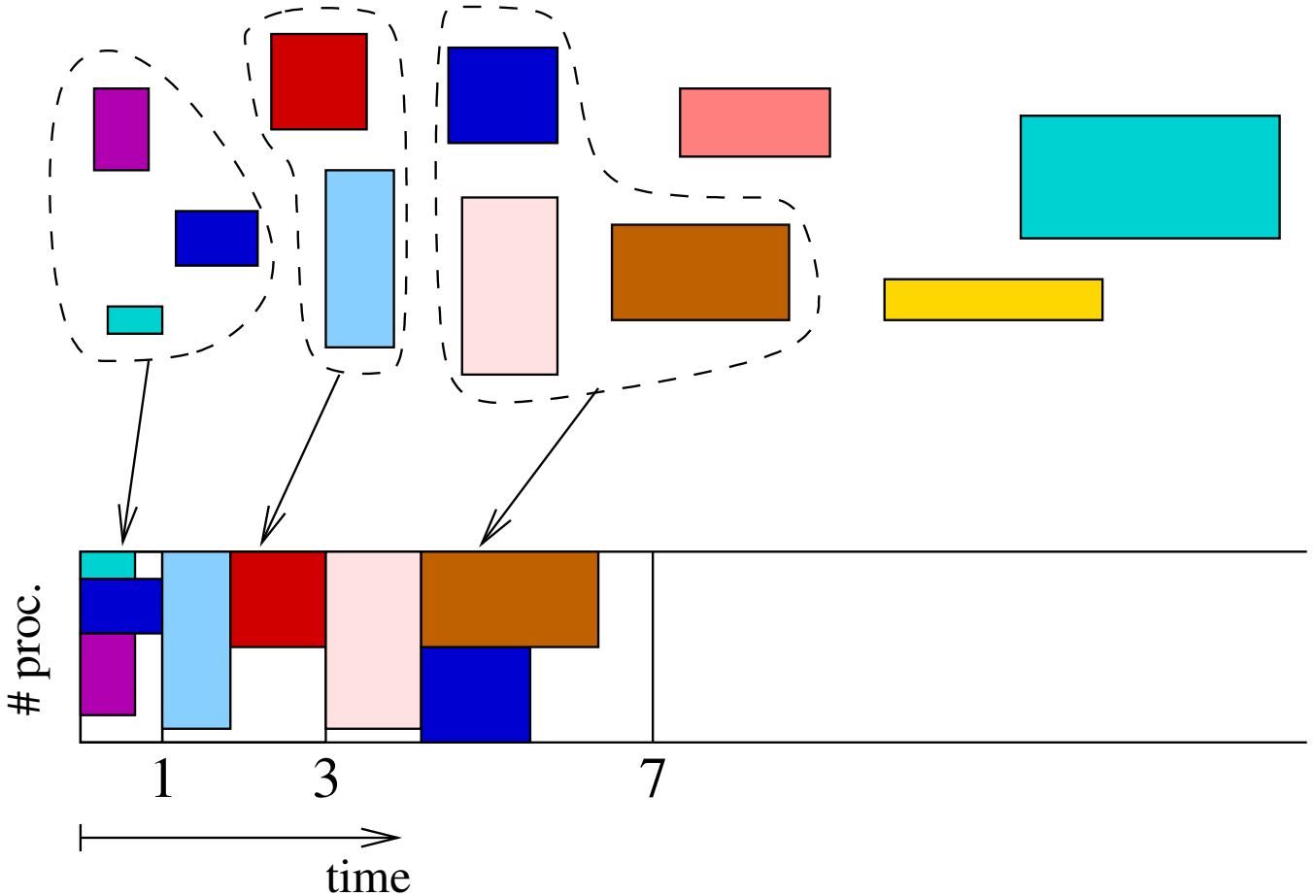
Un dessin vaut mieux que tous les mots



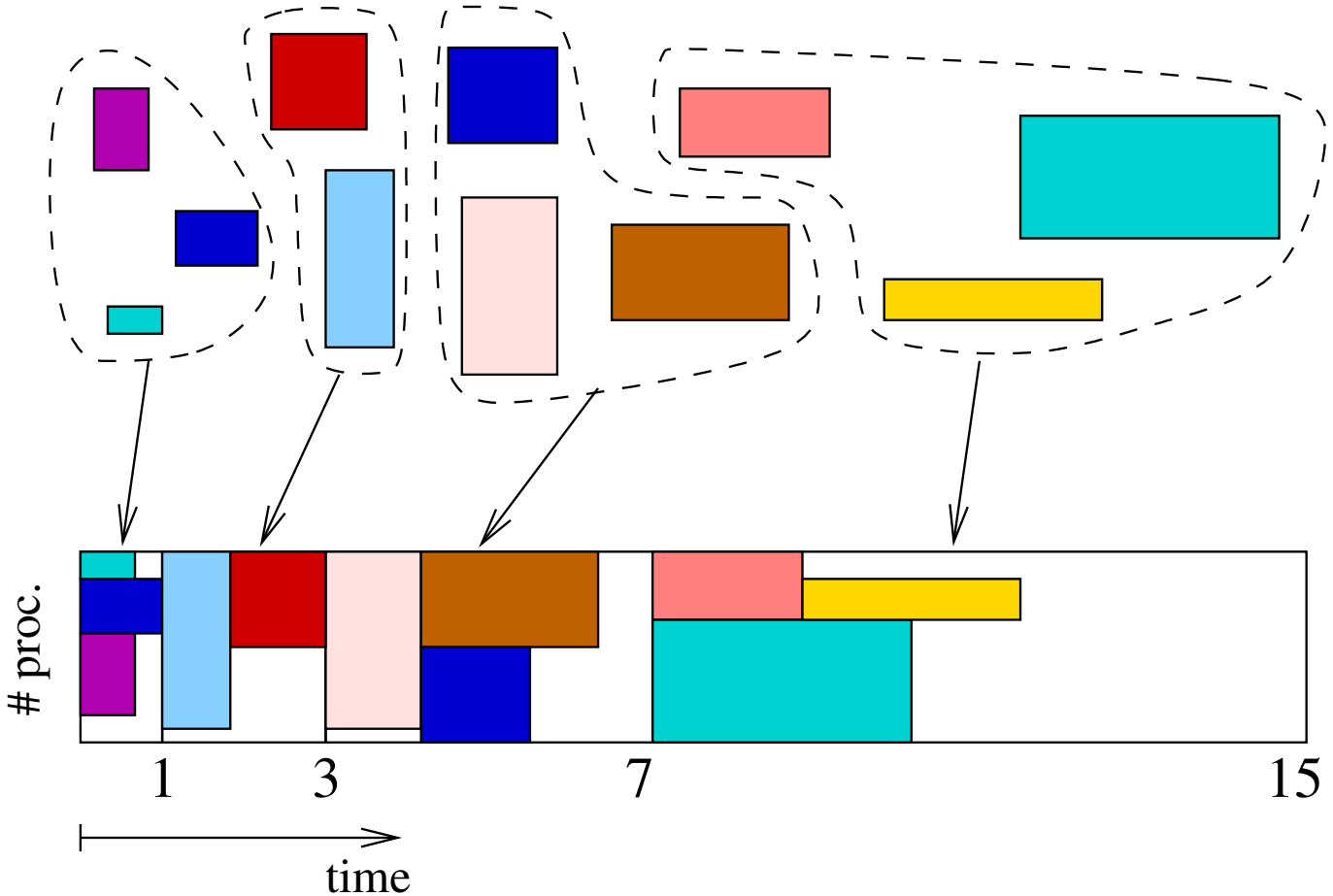
Un dessin vaut mieux que tous les mots



Un dessin vaut mieux que tous les mots



Un dessin vaut mieux que tous les mots



Les pieds sur terre

Algorithme garanti, mais :

- Spécifique aux tâches modelables
- Sélection : algorithme dynamique de haute complexité

Pour simplifier :

- Remplacer les batchs par des étagères
  - Sélection : un simple algorithme de sac-à-dos
  - Bonne propriété : sélection à une seule dimension
- Pas de garantie, mais une bonne validation expérimentale

Les pieds sur terre

Algorithme garanti, mais :

- Spécifique aux tâches modelables
- Sélection : algorithme dynamique de haute complexité

Pour simplifier :

- Remplacer les batchs par des étagères
  - Sélection : un simple algorithme de sac-à-dos
  - Bonne propriété : sélection à une seule dimension
- Pas de garantie, mais une bonne validation expérimentale

Mais alors, pourquoi pas dans OAR ?

Plan de l'exposé (rappel)

1. Un algorithme théorique
   - Modèle : tâches modelables
   - Un schéma en étagères
2. Le modèle d'ordonnancement dans OAR
   - Du point de vue utilisateur
   - Conséquences
3. Validation

Interface

Exemples :

```
oarsub -l nodes=3 blabla.sh

oarsub -l nodes=10,weight=1,walltime=2:00:00 script.sh

oarsub -p 'switch=1 AND memory > 150' script2.sh

oarsub -r '23/03/06 10 :00 :00' -l nodes=15 demo.sh
```

Reprenons depuis le début

- m machines homogènes
- n tâches modelables indépendantes :
  - $p_i(q)$: temps de calcul de la tâche i sur q machines
  - monotonie : pas d'accélération super-linéaire
  - r_i : date d'arrivée (soumission) de la tâche i
  - une priorité ω_i
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Makespan : $C_{\max} = \max_i C_i$
  - Temps de complétion moyen : $\sum_i \omega_i C_i$

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches modelables indépendantes :
  - $p_i(q)$: temps de calcul de la tâche i sur q machines
  - monotonie : pas d'accélération super-linéaire
  - r_i : date d'arrivée (soumission) de la tâche i
  - une priorité ω_i
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Makespan : $C_{\max} = \max_i C_i$
  - Temps de complétion moyen : $\sum_i \omega_i C_i$

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  - une priorité ω_i
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Makespan : $C_{max} = \max_i C_i$
  - Temps de complétion moyen : $\sum_i \omega_i C_i$

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  -
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Makespan : $C_{max} = \max_i C_i$
  - Temps de complétion moyen : $\sum_i \omega_i C_i$

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  - possibilité de réserver à l'avance
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Makespan : $C_{max} = \max_i C_i$
  - Temps de complétion moyen : $\sum_i \omega_i C_i$

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  - possibilité de réserver à l'avance
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Contentement des utilisateurs

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  - possibilité de réserver à l'avance
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Contentement des utilisateurs
  - Compréhension des utilisateurs

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  - possibilité de réserver à l'avance
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Contentement des utilisateurs
  - Compréhension des utilisateurs

Plus :

- Notion de propriétés : proche de ressources dédiées

Reprenons depuis le début

- m machines multi-processeurs et différentes
- n tâches rigides indépendantes :
  - w_i processeurs sur q_i machines pour un temps au plus p_i
  - r_i : date d'arrivée (soumission) de la tâche i
  - possibilité de réserver à l'avance
- Exécution exclusive et non préemptible
- Optimisation bi-critère :
  - Contentement des utilisateurs
  - Compréhension des utilisateurs

Plus :

- Notion de propriétés : proche de ressources dédiées
- Horizon infini

Que sait-on faire ?

- "Pseudo"-clairvoyant : walltime = mauvaise approximation
- Comment parler de C_i en horizon infini ?
- Réservations : aucune étude théorique pour tâches parallèles
- Ressources dédiées, pas hétérogènes en vitesse

Que sait-on faire ?

- "Pseudo"-clairvoyant : walltime = mauvaise approximation
- Comment parler de C_i en horizon infini ?
- Réservations : aucune étude théorique pour tâches parallèles
- Ressources dédiées, pas hétérogènes en vitesse

Dans OAR :

- "Premier arrivé, premier servi"
- Allocation gloutonne des ressources

Adaptations

Horizon "fini" :

- retour au temps zéro tous les jours (8h, 13h)
- plus de réactivité la journée, grands jobs le soir

Gestion simple des réservations :

- blocage des processeurs sur toute l'étagère
- la sélection reste sur une seule dimension
- se repose sur la compaction pour améliorer l'utilisation

Les propriétés ?

Ressources dédiées :

- Complexifie le problème de sélection dans le cas général
- Mais souvent, instances simples

- Algorithme exact par programmation dynamique
- Retour à un algo glouton pour les instances complexes

Plan de l'exposé (rappel)

1. Un algorithme théorique
   - Modèle : tâches modelables
   - Un schéma en étagères
2. Le modèle d'ordonnancement dans OAR
   - Du point de vue utilisateur
   - Conséquences
3. Validation

Contexte

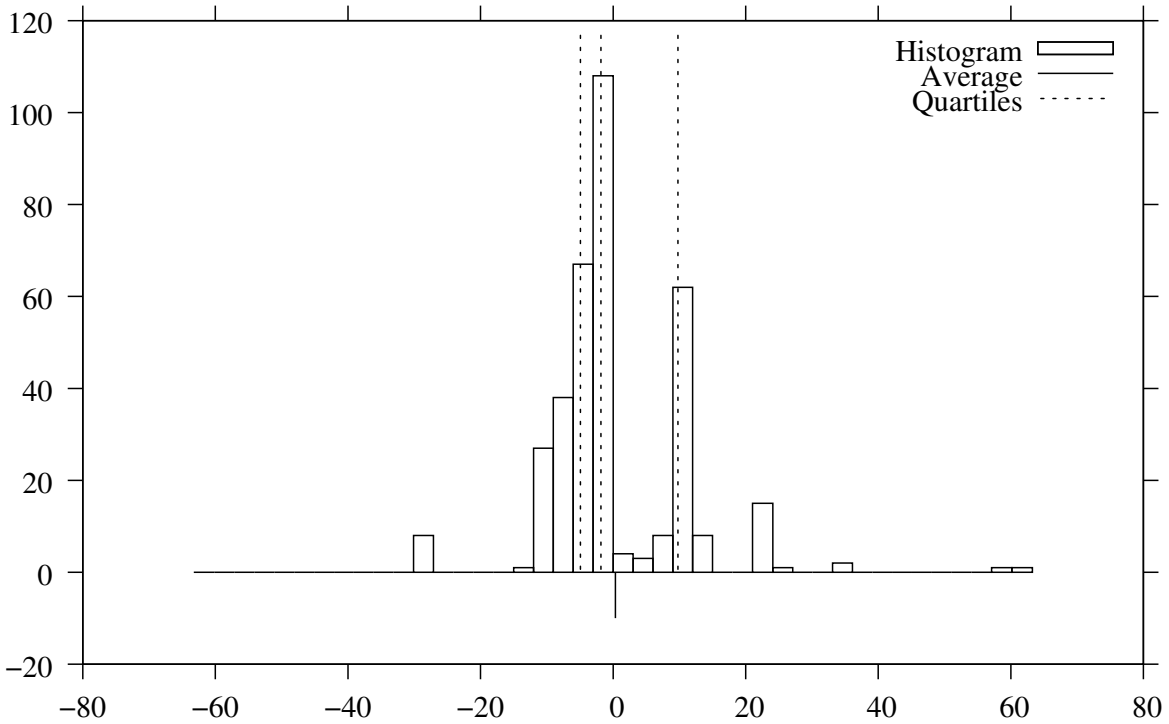
Simulation pour rejouer des traces

- Périodes "chargées" du I-Cluster2
- Comparaison avec le FCFS standard d'OAR
- Biais : les utilisateurs s'adaptent à l'environnement

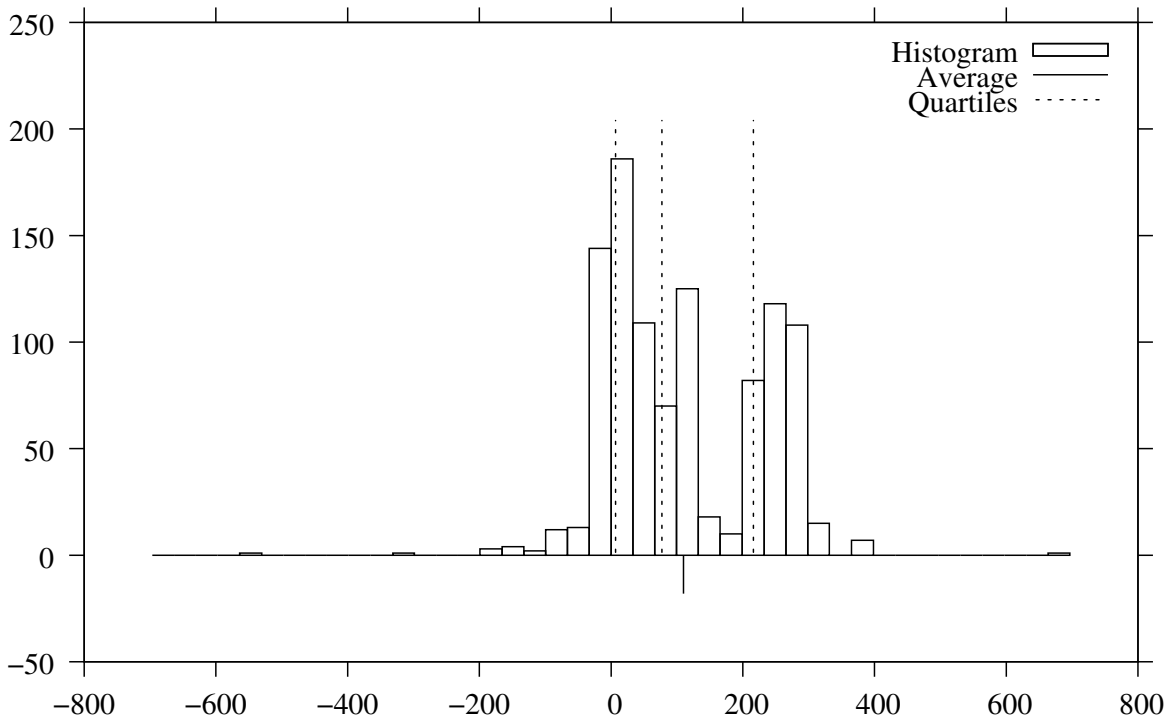
Résultats intéressants :

- Ordonnancements différents, mais résultats similaires
- Quelques instances plus "dures" sont mieux traitées

Exemple 1


Histogram
Average
Quartiles
120
100
80
60
40
20
0
−20
−80
−60
−40
−20
0
20
40
60
80

Exemple 2


Histogram
Average
Quartiles
250
200
150
100
50
0
−50
−800
−600
−400
−200
0
200
400
600
800

Conclusions

- Difficultés de mise en œuvre des algos théoriques
- Problème d'adéquation des modèles
- Soulève quelques problèmes qu'il serait intéressant d'étudier

- Mais c'est possible
- Reste à le faire accepter dans OAR