



# Résolution de conflits par algorithmes stochastiques parallèles

Xavier OLIVE

SUPAERO — M2R SLCP

Laboratoire d'Optimisation Globale (ENAC)

Toulouse, de février à juillet 2006

7 septembre 2006



## Introduction

- Les problèmes de résolution de conflits aériens étant **fortement combinatoires**, les méthodes déterministes échouent sur des problèmes de grande taille.
- Les **algorithmes génétiques** ayant fait leurs preuves, l'objet de l'étude a été d'éprouver une métaheuristique semblable, s'inspirant du **comportement des fourmis**.
- Afin de gagner en robustesse, fiabilité et rapidité, on s'est ensuite attaché à **paralléliser l'algorithme** sur un cluster de machines, selon différentes méthodes.

## Introduction

## Le problème et l'algorithme

○○  
○○  
○○  
○○○○

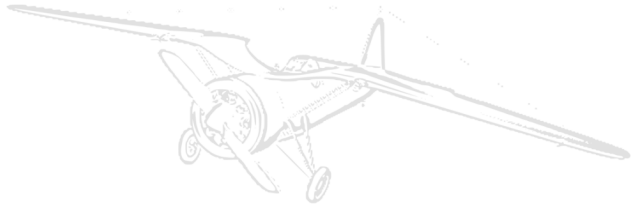
## Modèles parallèles

○○  
○○○○  
○○  
○○○

## Performances

○  
○  
○

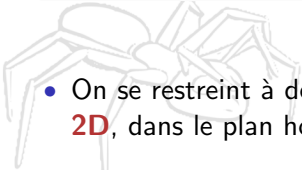
## Conclusion

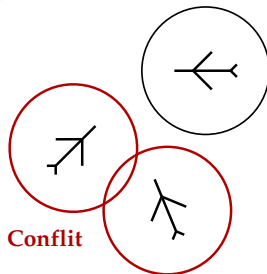




## Le problème de résolution de conflits

Connaissant les positions et vitesses des avions à un instant  $t$ ,  
**quelles sont les manœuvres à donner à ces avions** afin  
que les trajectoires ne génèrent aucun conflit tout en  
engendrant un retard minimal ?

- 
- On se restreint à des **trajectoires 2D**, dans le plan horizontal.
  - La distance horizontale de sécurité est de **5 miles nautiques**.





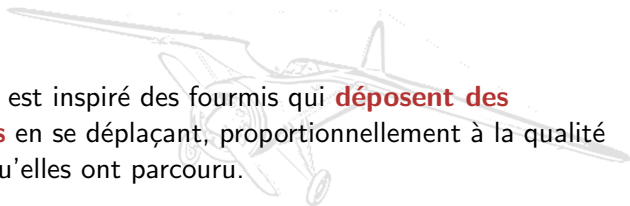
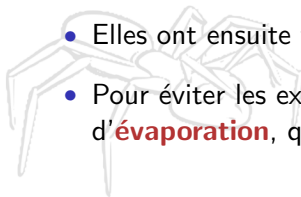
## Le problème de résolution de conflits

- Au sein d'un cluster de  $n$  avions, le domaine des solutions contient  $2^{\frac{1}{2} \cdot n(n-1)}$  composantes connexes : **le problème est fortement combinatoire**.
- On peut utiliser différentes méthodes pour résoudre ce problème :
  - **déterministes** :  $A^*$ , programmation par contraintes ;
  - **stochastiques** : algorithmes génétiques.
- On s'intéresse ici aux algorithmes de **colonies de fourmis**.



## Les algorithmes de colonies de fourmis

- L'algorithme est inspiré des fourmis qui **déposent des phéromones** en se déplaçant, proportionnellement à la qualité du chemin qu'elles ont parcouru.
- Elles ont ensuite tendance à **suivre ces phéromones**.
- Pour éviter les extrema locaux, on rajoute un phénomène d'**évaporation**, qui tend à faire disparaître les phéromones.





# Les algorithmes de colonies de fourmis

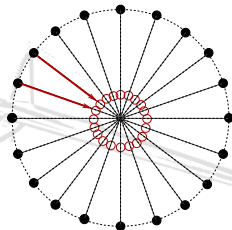
sur un problème de voyageur de commerce

- Des fourmis **parcourent le graphe**, en construisant un trajet complet. Le choix de la ville suivante est influencé par les phéromones déposées à l'étape précédente.
- Ensuite, elles **déposent des phéromones** sur le chemin qu'elles ont parcouru :  $\Delta\tau_{ij}(t) \propto \frac{1}{\sum L_{ij}}$
- À chaque itération, on **évapore** les pistes :  $\tau_{ij} \leftarrow \rho \cdot \tau_{ij}$

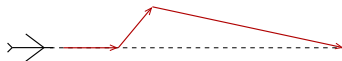


## Modélisation du problème de résolution de conflits

- On veut résoudre le problème de  $n$  avions, avançant tous à la même vitesse, **placés sur un cercle** et pointant vers le centre du cercle.

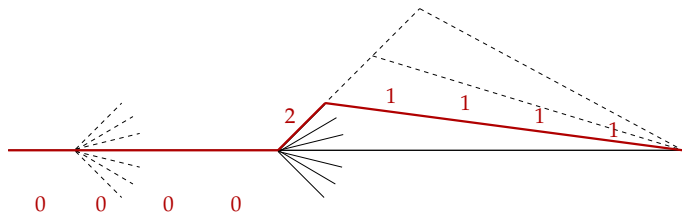


- On **discrétise les trajectoires**.
- Afin de minimiser les manœuvres, on n'autorise qu'**un seul écart** par rapport à la ligne droite :





## Modélisation du problème de résolution de conflits



- On **évalue chaque trajectoire** sans conflit :
  - 0 pour un tronçon **en ligne droite** ;
  - 2 pour un tronçon qui **s'éloigne** de la ligne droite ;
  - 1 pour un tronçon qui **retourne** vers la ligne droite .
- La trajectoire ci-dessus est évaluée à 6.



# Modélisation du problème de résolution de conflits

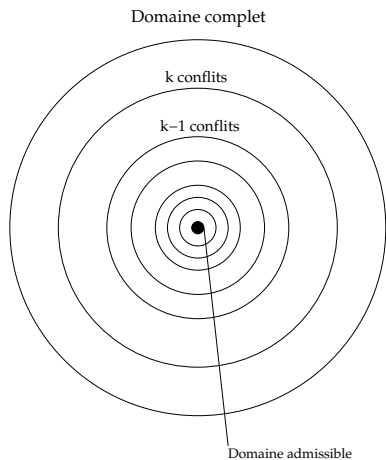
## Réduction du domaine de recherche

- Si on autorise 6 angles d'écart par rapport à la ligne droite, pour un problème à 30 avions avec une précision de 20 pas de temps, on a  **$240^{30}$  pistes de phéromones** à tenir à jour.
- Au lieu de chercher une solution du problème complet, une fourmi cherchera une solution à la **trajectoire d'un seul avion**, sans avoir de vision globale du problème.
- Chaque fourmi a 240 pistes de phéromones à tenir à jour. Le domaine est alors réduit à  **$30 \times 240$  pistes de phéromones**.



# Modélisation du problème de résolution de conflits

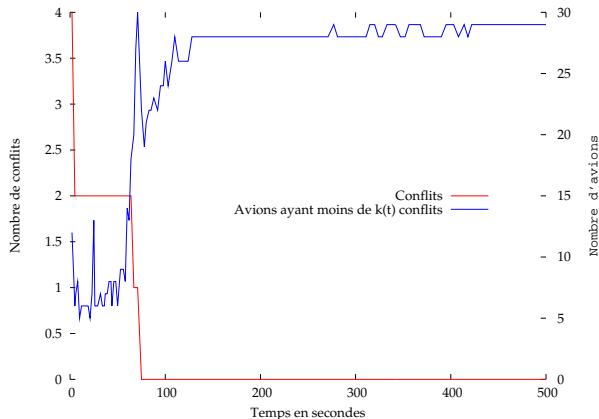
## Relaxation de contraintes



- En tolérant des contraintes, on arrive **cibler le domaine de recherche**.
- Si  $\frac{30}{k}$  avions ont une trajectoire à  $k$  conflits, on **renforce la contrainte** :  $k \leftarrow k - 1$ .



# Modélisation du problème de résolution de conflits



Problème d'**instabilité des résultats**.

Introduction

Le problème et l'algorithme

○○  
○○  
○○○○

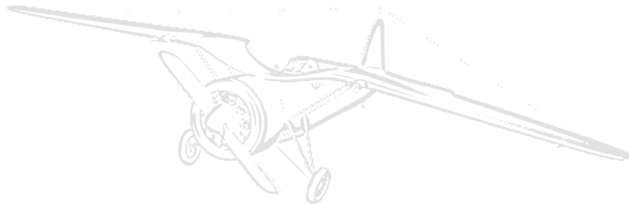
Modèles parallèles

○○  
○○○○  
○○  
○○○

Performances

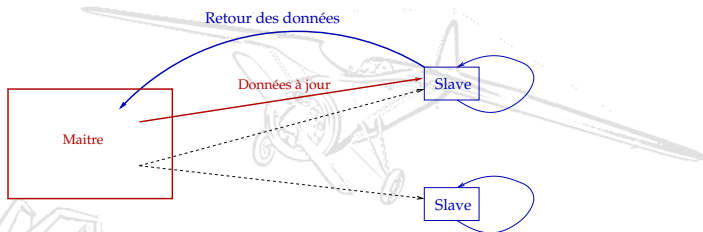
○  
○  
○

Conclusion





## Un modèle maître-esclave

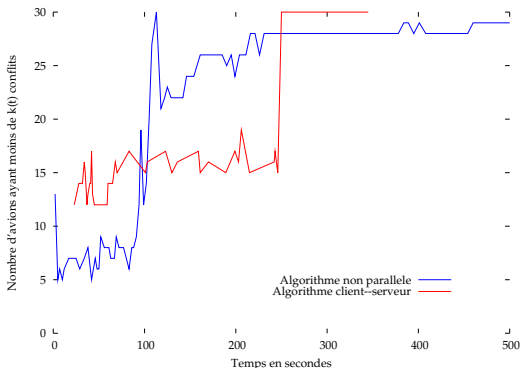


- Afin d'augmenter la population de fourmis, on **déporte les opérations** vers des machines esclaves.
- Le maître concatène les résultats des esclaves et **fournit des pistes de phéromones à jour**.



# Modèle maître-esclave

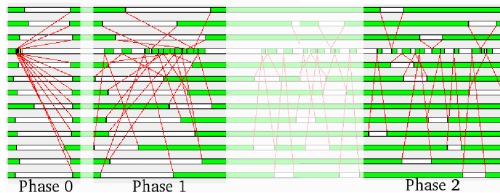
Résultats pour 13 machines,  $20 \times 30$  fourmis.





# Un modèle maître–esclave

## Optimisations

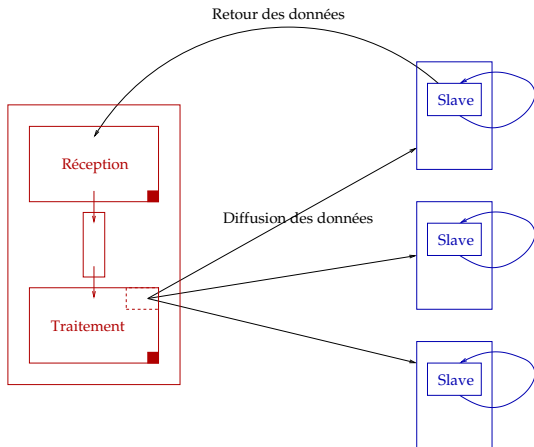


On pourra envisager, afin d'optimiser les communications :

- **multithreading** du maître, séparation des tâches de communication et de calcul ;
- **asynchronisation** des esclaves, poursuite des itérations avec des données obsolètes.

# Un modèle maître-esclave

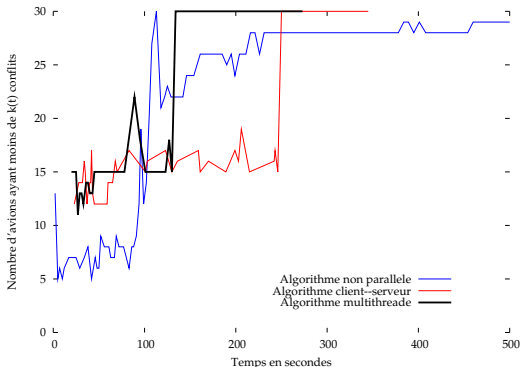
## Optimisations





# Modèle maître-esclave

## Résultats

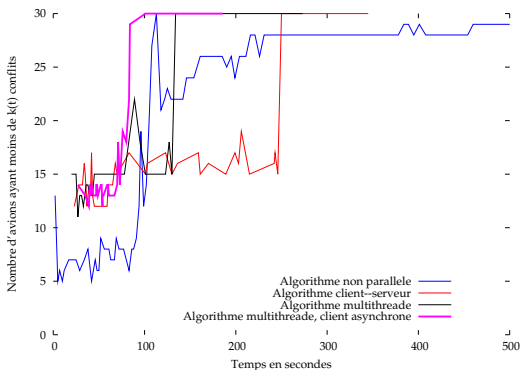


Le maître a **deux processus légers.**



# Modèle maître-esclave

## Résultats

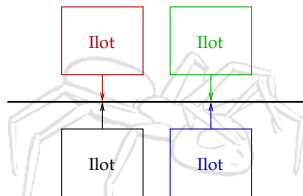


Les esclaves continuent d'itérer avec des **données obsolètes**.



## Un modèle par îlots

- Le principe des îlots existe déjà pour les algorithmes génétiques.



- On espère ainsi **éviter les extrema locaux**.

- Chaque machine lance le **même algorithme avec une graine différente**.

- La méthode comprend une phase où chaque îlot **partage une infime partie des données**.



# Modèle par îlots

## Résultats

- La **méthode n'est pas satisfaisante** pour ce type de problème : le comportement est semblable à celui de l'algorithme local.
- Le **domaine parcouru est très grand** par rapport au domaine de solutions admissibles : le domaine parcouru n'est pas assez restrictif.



## Un modèle mixte

Un nouveau modèle a été créé pour :

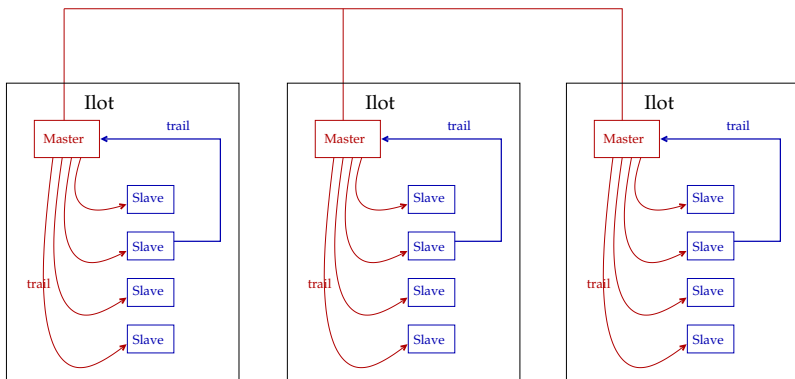
- réduire le domaine aux **solutions admissibles**. (modèle maître-esclave) ;
- ne pas rester coincé dans des **minima locaux** (modèle par îlots).

On réalise des **îlots de maîtres**, tous identiques, qui communiquent entre eux une partie des données, et qui délèguent leurs calculs à des esclaves sous leurs ordres.



# Un modèle mixte

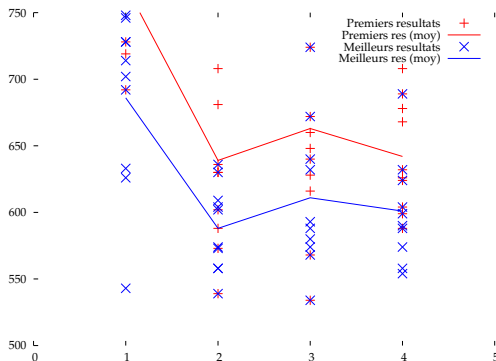
un seul chemin





# Modèle mixte

## Résultats



On trouve les meilleurs résultats avec **2 îlots de 6 machines**.

Introduction

Le problème et l'algorithme

○○  
○○  
○○○○

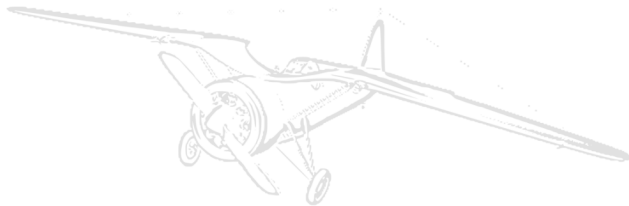
Modèles parallèles

○○  
○○○○  
○○  
○○○

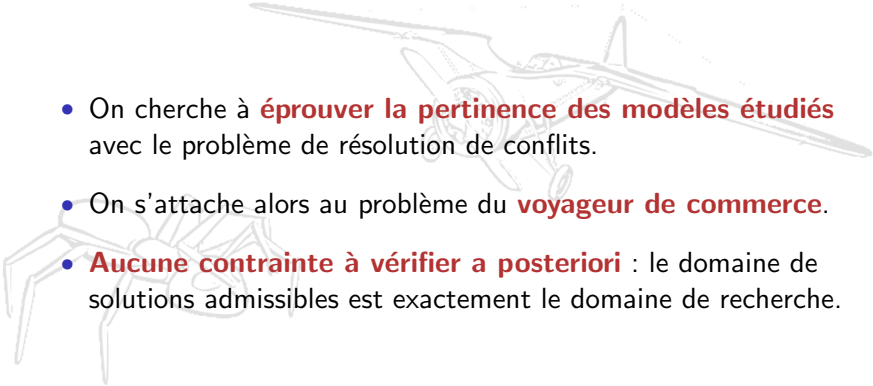
Performances

○  
○  
○

Conclusion



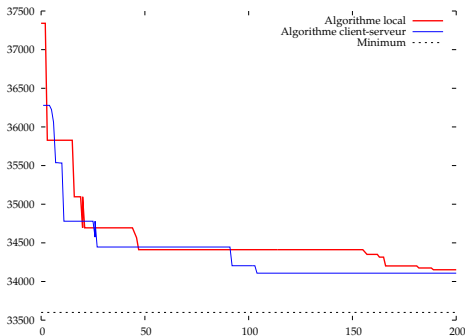
## Voyageur de commerce

- 
- On cherche à **éprouver la pertinence des modèles étudiés** avec le problème de résolution de conflits.
  - On s'attache alors au problème du **voyageur de commerce**.
  - **Aucune contrainte à vérifier a posteriori** : le domaine de solutions admissibles est exactement le domaine de recherche.



# Voyageur de commerce

## Modèle maître-esclave

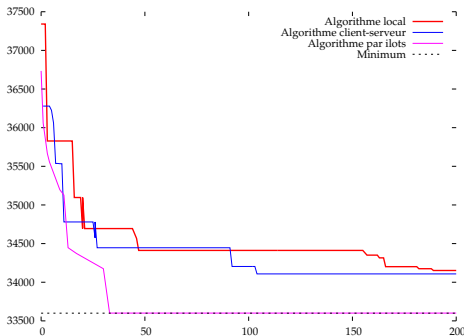


- Le modèle maître-esclave est trop sélectif pour éviter les **minima locaux**.



# Voyageur de commerce

## Modèle par îlots



- Le **modèle par îlots** diversifie correctement les solutions proposées et fait converger vers la **solution optimale**.



## Conclusion

- Les algorithmes à colonies de fourmis peuvent se révéler **efficaces sur différents problèmes.**
  - La parallélisation apporte :
    - **robustesse** de la convergence ;
    - **qualité** des résultats ;
    - **diversité** dans le parcours du domaine de recherche.
  - Les meilleures méthodes de parallélisation restent des **méthodes ad hoc.**
- 