Cours de Master Recherche
Spécialité CODE :
Résolution de problèmes combinatoires

Christine Solnon

LIRIS, UMR 5205 CNRS / Université Lyon 1

2007

Rappel du plan du cours

16 heures de cours

- 1 - Introduction ($\sim$ 4 heures)
  - Qu'est-ce qu'un problème complexe ?
  - Exemples de problèmes complexes
- 2 - Approches complètes ($\sim$ 4 heures)
  - Structuration de l'espace de recherche en arbre
  - Techniques d'élagage et heuristiques d'ordre
- 3 - Approches incomplètes ($\sim$ 8 heures)
  - Approches basées sur le voisinage
  - Approches constructives

10 heures de TP + 4 heures d'exposés $\rightsquigarrow$ Carole Knibbe

Résolution du problème SAT par recherche de cliques

- Choix d'une méthode de résolution
  $\rightsquigarrow$ implémentation et expérimentation
- Rédaction d'un article et exposé oral

Approches constructives

Principe général

- Approches constructives :
  ⇝ constructions incrémentales de combinaisons
- Utilisation d'un "modèle"
  ⇝ heuristique de choix du composant à ajouter

Modèles statiques vs dynamiques

- Modèles statiques
  ⇝ Algorithmes gloutons (aléatoires)
- Modèles dynamiques
  ⇝ modifier le modèle / expérience passée
  - Algorithmes par estimation de distribution
    ⇝ statistiques sur les expériences passées
  - Algorithmes à base de colonies de fourmis
    ⇝ expérience passée compilée sous forme de phéromone

Principe général des algorithmes gloutons

Construction incrémentale d'une combinaison $e \in E$

- $e \leftarrow$ combinaison vide
- *Cand* $\leftarrow$ ensemble des composants de combinaisons
- tant que *Cand* $\neq \emptyset$ faire
  - Choisir le meilleur composant de *Cand* et l'ajouter à e
    $\rightsquigarrow$ Heuristique de choix $\ll$ gloutonne $\gg$
  - Mettre à jour l'ensemble *Cand* des composants pouvant être ajoutés à e
- Retourner e

Pour certains problèmes, algorithme optimal
$\rightsquigarrow$ Simplex/problèmes linéaires, Dijkstra/plus court chemin

Comment concevoir un algorithme glouton ?

- Soit un problème d'optimisation (E, f)
  - $\rightsquigarrow$ E = ensemble des combinaisons candidates
  - $\rightsquigarrow$ f = fonction objectif à maximiser
- 3 points à définir :
  1. Identifier les composants des combinaisons
  2. Définir une fonction mettant à jour l'ensemble des candidats par rapport à une combinaison partielle
  3. Définir une fonction heuristique évaluant la qualité d'un composant par r apport à une combinaison partielle

Exemple 1 : Le voyageur de commerce

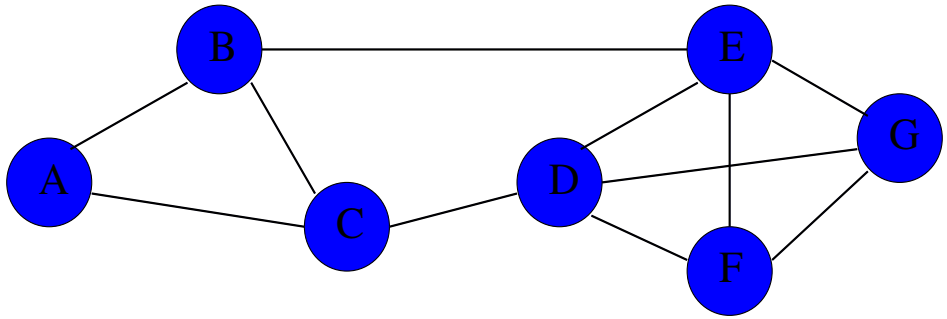
- Choisir aléatoirement un sommet $s_i \in S$
- $\pi =< s_i >$
- $Cand \leftarrow S - \{s_i\}$
- tant que $Cand \neq \emptyset$ faire
  - Choisir $s_j \in$ candidats selon une heuristique gloutonne
  - ajouter s_j à la fin de π
  - $Cand \leftarrow Cand - \{s_j\}$

Quelle heuristique gloutonne ???

Exemple 2 : Recherche d'une clique maximum

- $C \leftarrow \emptyset$
- $Cand \leftarrow S$
- tant que $Cand \neq \emptyset$ faire
  - Choisir $s_j \in$ candidats selon une heuristique gloutonne
  - $C \leftarrow C \cup \{s_j\}$
  - $Cand \leftarrow Cand \cap \{s_i / (s_i, s_j) \in A\}$

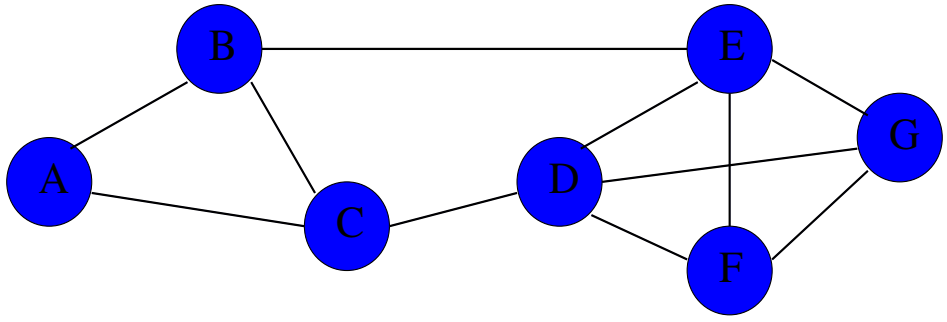
Quelle heuristique gloutonne ???



Exemple 3 : Coloriage d'un graphe

- $nbCoul \leftarrow 0$
- $Cand \leftarrow S$
- tant que $Cand \neq \emptyset$ faire
  - Choisir $s_j \in$ candidats selon une heuristique gloutonne
  - Si $\exists i \leq nbCoul$ tq $\forall s_k$ adjacent à $s_j, couleur(s_k) \neq i$
    Alors $couleur(s_j) \leftarrow i$
    Sinon $nbCoul \leftarrow nbCoul + 1$; $couleur(s_j) \leftarrow nbCoul$
  - $Cand \leftarrow Cand - \{s_j\}$

Quelle heuristique gloutonne ???



Exemple 4 : Résoudre un CSP $(X, D, C) \rightsquigarrow$ MaxCSP

- $A \leftarrow \emptyset$
- $Cand \leftarrow X$
- pour toute variable $X_i \in X$, $D_{\text{filtré}}(X_i) \leftarrow D(X_i)$
- tant que $Cand \neq \emptyset$ faire
  - Choisir une variable $X_j \in Cand$
  - Choisir une valeur $v \in D(X_j)$
  - $A \leftarrow A \cup < X_j, v >$
  - $Cand \leftarrow Cand - \{X_j\}$

Quelle heuristique gloutonne pour choisir la variable ???
Quelle heuristique gloutonne pour choisir la valeur ???

Gloutons aléatoires (1)

Problème

En général, un algorithme glouton retourne une assez bonne solution ... mais (à peu près) toujours la même

Idée

- Introduire un peu d'aléatoire pour diversifier la recherche
- Exécuter plusieurs fois l'algorithme... et retourner la meilleure solution

Gloutons aléatoires (2)

Approches pour « introduire un peu d'aléatoire »

Approche "1 parmi les meilleurs"

- Sélectionner les k meilleurs composants
  ...ou bien les composants à k% de l'optimum
- Choisir au hasard 1 de ces composants

$\rightsquigarrow$ k = paramètre pour « régler » le degré d'aléatoire

Approche probabiliste

Sélectionner le prochain composant selon une probabilité

- Probabilité de choisir $c_i \in Cand$:
  $p(c_i) = \frac{h(c_i)^\alpha}{\sum_{c_k \in Cand} h(c_k)^\alpha}$
- Tirer un nombre flottant aléatoire x compris entre 0 et 1
- Choisir c_i tq $\sum_{j<i} p(c_j) < x \leq \sum_{j \leq i} p(c_j)$

Différentes approches basées sur les modèles (rappel)

- Modèle statique

  $\rightsquigarrow$ Algorithmes gloutons (aléatoires)

- Modèle dynamique

  $\rightsquigarrow$ modifier le modèle / expérience passée

  - Algorithmes par estimation de distribution
    $\rightsquigarrow$ statistiques sur les expériences passées
  - Algorithmes à base de colonies de fourmis
    $\rightsquigarrow$ expérience passée “compilée” sous forme de phéromone

Algorithmes par estimation de distribution (EDA)

Principe général

- Génération d'une population initiale $P(0)$
- $t \leftarrow 0$
- Tant que solution optimale non trouvée et $t < t_{max}$ faire :
  - Sélectionner un ensemble $S(t) \subseteq P(t)$ de solutions « prometteuses »
  - Construire un modèle probabiliste $M(t)$ à partir de $S(t)$
  - Générer de nouvelles solutions $N(t)$ à partir de $M(t)$
  - Créer une nouvelle population $P(t+1) \subseteq S(t) \cup N(t)$
  - $t \leftarrow t+1$

Approche évolutionniste
$\rightsquigarrow$ population générée à partir d'un modèle qui évolue

Algorithmes par estimation de distribution (EDA)

Choix d'un modèle probabiliste

$\rightsquigarrow$ compromis à trouver entre simplicité et qualité

- PBIL = Population Based Incremental Learning [Baluja 1994]
  - $\rightsquigarrow$ vecteur associant une proba à chaque couple variable/valeur
  - $\rightsquigarrow$ ne prend pas en compte les dépendances entre variables
- COMIT [Baluja et Davies 1997]
  - $\rightsquigarrow$ réseau bayésien structuré en arbre
  - $\rightsquigarrow$ prise en compte de dépendances binaires
- BOA [Pelikan, Goldberg et Cantú-paz 2000]
  - $\rightsquigarrow$ réseau bayésien
- ...

Pour en savoir plus :
`http://www.iba.k.u-tokyo.ac.jp/english/EDA.htm`

Différentes approches basées sur les modèles (rappel)

- Modèle statique

  $\rightsquigarrow$ Algorithmes gloutons (aléatoires)

- Modèle dynamique

  $\rightsquigarrow$ modifier le modèle / expérience passée

  - Algorithmes par estimation de distribution
    $\rightsquigarrow$ statistiques sur les expériences passées
  - Algorithmes à base de colonies de fourmis
    $\rightsquigarrow$ expérience passée “compilée” sous forme de phéromone

Un peu d'éthologie pour commencer...

Auto-organisation des insectes sociaux

Emergence d'un comportement au niveau global, en réponse à des interactions multiples entre composants de plus bas niveau

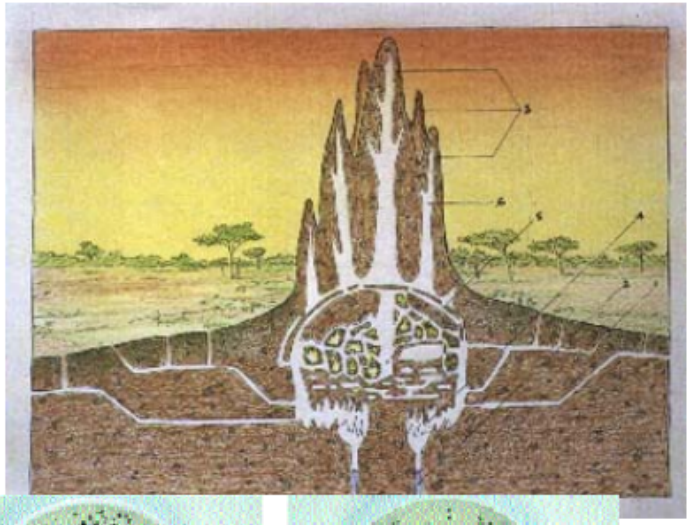
- Emergence $\rightsquigarrow$ comportement global non "programmé"
- Comportement $\rightsquigarrow$ structure spatio-temporelle
- Interactions $\rightsquigarrow$ communications directes ou indirectes
  - Communication indirecte $\rightsquigarrow$ Stigmergie [Grassé 59] : modifier l'environnement / travail en cours
    $\rightsquigarrow$ Rétroactions (feedback) positives et négatives

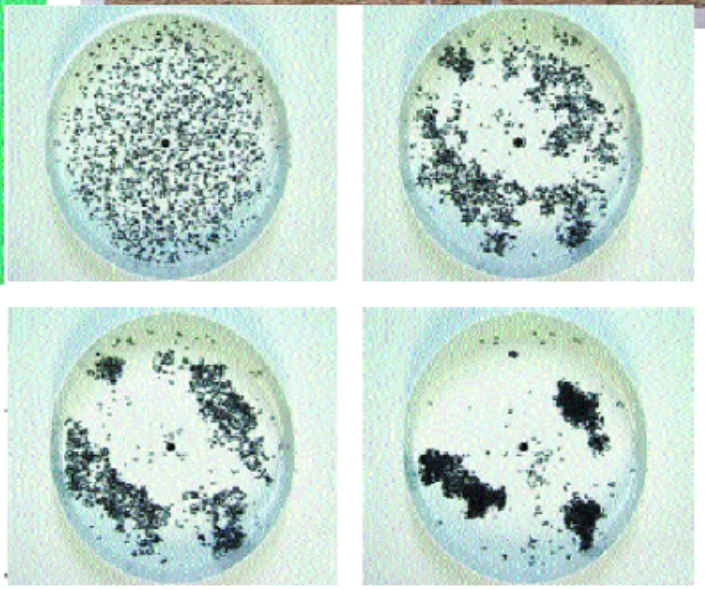
Quelques caractéristiques des systèmes auto-organisés

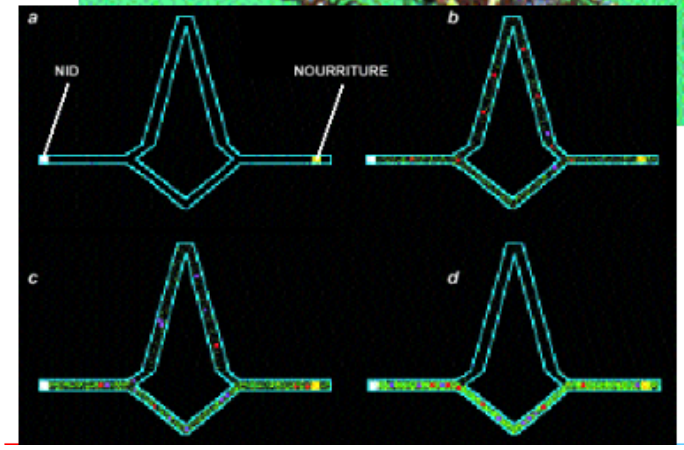
- Dynamisme
- Décentralisation $\rightsquigarrow$ Robustesse et Flexibilité
- Capacité à produire des modèles stables

... les systèmes auto-organisés sont des systèmes complexes

Exemples de systèmes auto-organisés





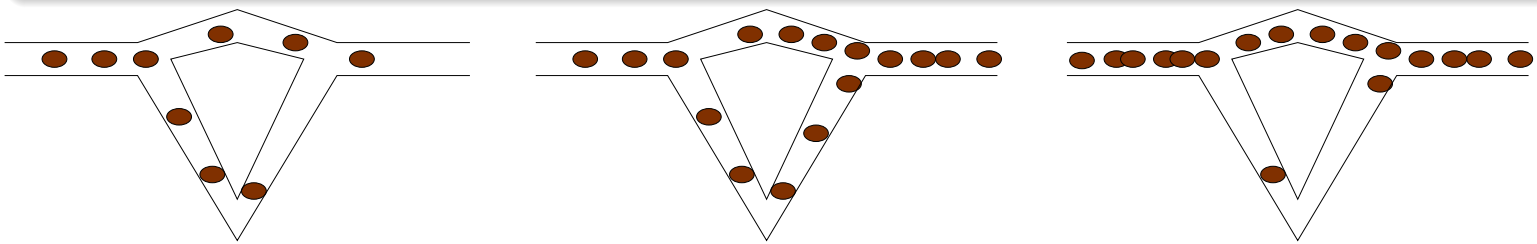


Optimisation par colonies de fourmis

Inspiration : recherche de nourriture par une colonie de fourmis

Individus simples et autonomes
\+ Communication indirecte à travers l'environnement
- Dépôt de traces de phéromone
- Déplacement aléatoire, guidé par les traces
- Evaporation/diffusion de la phéromone

$\Longrightarrow$ Emergence de « plus courts chemins »



- Au début : absence de trace $\rightsquigarrow$ équiprobabilité
- Les fourmis prenant le + court chemin reviennent + vite
  $\Rightarrow$ les traces sur le + court chemin augmentent
- Autocatalyse : dépôt de phéromone $\Rightarrow$ augmentation de la probabilité $\Rightarrow$ dépôt de phéromone $\Rightarrow$ ...
- Evaporation : limite l'autocatalyse

Optimisation par colonies de fourmis

Modélisation mathématique [Deneubourg et al.90]

- Modèle stochastique de la dynamique d'une colonie de fourmis
- Identifier les mécanismes permettant aux fourmis de s'auto-organiser pour résoudre le problème de la recherche d'un plus court chemin
  - La probabilité de choix d'un chemin dépend de la quantité de phéromone déposée ... qui elle même dépend du nombre de fourmis précédemment passées par ce chemin
  - Evaporation de la phéromone
    ... pas indispensable pour trouver un plus court chemin !

Naissance d'une méta-heuristique

Premier algorithme à base de fourmis : Ant System

proposé par M. Dorigo en 1992 pour le voyageur de commerce

Successeurs de Ant System

Ant Colony System [Dorigo & Gambardella 97],
$\mathcal{MAX} - \mathcal{MIN}$ AS [Stützle & Hoos 00],
Hyper-cube AS [Blum, Roli & Dorigo 01], ...

Très nombreuses applications

Problèmes de routage de véhicules, d'affectation quadratique, de coloriage de graphes, d'emplois du temps, ...

Généralisation

Méta-heuristique Ant Colony Optimization (ACO)

Résolution de problèmes avec ACO

Définir une structure phéromonale

Ensemble de composants sur lesquels sera déposée la phéromone

Recherche de solutions par des fourmis artificielles

- Comportement inspiré des fourmis réelles...

  Communication indirecte par dépôt de phéromone
  Décision aléatoire, guidée par la phéromone
  Evaporation de la phéromone

- ... avec des capacités supplémentaires

  Dépôt de phéromone retardé et proportionnel à la qualité,
  Utilisation d'heuristiques locales, de recherche locale, ...

ACO : Principe général

- initialisation des traces de phéromone
- répéter
  1. chaque fourmi construit une solution
  2. mise-à-jour des traces de phéromone
- jusqu'à solution optimale trouvée ou stagnation

ACO : Principe général

- initialisation des traces de phéromone
- répéter
  1. **chaque fourmi construit une solution**
  2. mise-à-jour des traces de phéromone
- jusqu'à solution optimale trouvée ou stagnation

Construction gloutonne aléatoire d'une solution par une fourmi

- Soit $\mathcal{C}$ = début de solution et *cand* = candidats
- Choisir $v_j \in cand$ selon la probabilité

$$p(v_j) = \frac{[\tau_{\mathcal{C}}(v_j)]^{\alpha}.[\eta_{\mathcal{C}}(v_j)]^{\beta}}{\sum_{v_k \in cand}[\tau_{\mathcal{C}}(v_k)]^{\alpha}.[\eta_{\mathcal{C}}(v_k)]^{\beta}}$$

ACO : Principe général

- initialisation des traces de phéromone
- répéter
  1. **chaque fourmi construit une solution**
  2. mise-à-jour des traces de phéromone
- jusqu'à solution optimale trouvée ou stagnation

Construction gloutonne aléatoire d'une solution par une fourmi

- Soit $\mathcal{C}$ = début de solution et *cand* = candidats
- Choisir $v_j \in cand$ selon la probabilité

$$p(v_j) = \frac{[\tau_{\mathcal{C}}(v_j)]^{\alpha}.[\eta_{\mathcal{C}}(v_j)]^{\beta}}{\sum_{v_k \in cand}[\tau_{\mathcal{C}}(v_k)]^{\alpha}.[\eta_{\mathcal{C}}(v_k)]^{\beta}}$$

$\tau_{\mathcal{C}}(v_j) \rightsquigarrow$ facteur phéromonal (expérience passée de la colonie)

ACO : Principe général

- initialisation des traces de phéromone
- répéter
  1. **chaque fourmi construit une solution**
  2. mise-à-jour des traces de phéromone
- jusqu'à solution optimale trouvée ou stagnation

Construction gloutonne aléatoire d'une solution par une fourmi

- Soit $\mathcal{C}$ = début de solution et *cand* = candidats
- Choisir $v_j \in cand$ selon la probabilité

$$p(v_j) = \frac{[\tau_{\mathcal{C}}(v_j)]^{\alpha}.[\eta_{\mathcal{C}}(v_j)]^{\beta}}{\sum_{v_k \in cand}[\tau_{\mathcal{C}}(v_k)]^{\alpha}.[\eta_{\mathcal{C}}(v_k)]^{\beta}}$$

$\eta_{\mathcal{C}}(v_j) \rightsquigarrow$ facteur heuristique (dépendant du problème)

ACO : Principe général

- initialisation des traces de phéromone
- répéter
  1. **chaque fourmi construit une solution**
  2. mise-à-jour des traces de phéromone
- jusqu'à solution optimale trouvée ou stagnation

Construction gloutonne aléatoire d'une solution par une fourmi

- Soit $\mathcal{C}$ = début de solution et *cand* = candidats
- Choisir $v_j \in cand$ selon la probabilité

$$p(v_j) = \frac{[\tau_{\mathcal{C}}(v_j)]^{\alpha}.[\eta_{\mathcal{C}}(v_j)]^{\beta}}{\sum_{v_k \in cand}[\tau_{\mathcal{C}}(v_k)]^{\alpha}.[\eta_{\mathcal{C}}(v_k)]^{\beta}}$$

α, $\beta \rightsquigarrow$ poids des facteurs (paramètres)

ACO : Principe général

- initialisation des traces de phéromone
- répéter
  1. chaque fourmi construit une solution
  2. **mise-à-jour des traces de phéromone**
- jusqu'à solution optimale trouvée ou stagnation

Mise-à-jour des traces de phéromone

- Evaporer : multiplier les traces de phéromone par $(1 - \rho)$
  $\rightsquigarrow$ ρ = facteur d'évaporation compris entre 0 et 1
- Récompenser : ajouter de la phéromone sur les composants des meilleures solutions proportionnellement à la qualité

Exemple : le voyageur de commerce

Structure phéromonale

Phéromone déposée sur les arêtes du graphe :
$\tau(v_i, v_j) \rightsquigarrow \ll$ désirabilité $\gg$ de visiter v_j juste après v_i

A chaque cycle, chaque fourmi construit un chemin

- Choix aléatoire du sommet de départ
- Probabilité, quand la fourmi est à v_i, d'aller à v_j :

$$p(v_j) = \frac{[\tau(v_i, v_j)]^{\alpha}.[1/d(v_i, v_j)]^{\beta}}{\sum_{v_k \in cand}[\tau(v_i, v_k)]^{\alpha}.[1/d(v_i, v_k)]^{\beta}}$$

où *cand* = sommets que la fourmi n'a pas déjà visités

Mise-à-jour de la phéromone

- Ajout de phéromone sur les arêtes du meilleur chemin
  (Quantité inversement proportionnelle à la longueur)
- Evaporation

Le point de vue « R.O. » sur ACO

ACO = Greedy Randomized Adaptive Search

- Greedy : Construction de solutions selon un principe glouton
- Randomized : Diversification en introduisant de l'aléatoire
- Adaptive : Adapter la recherche en fonction du passé
  - Capitaliser l'expérience passée par dépot de phéromone
  - Exploiter l'expérience passée en biaisant la recherche par rapport aux quantités de phéromone

Intensifier/diversifier la recherche avec ACO

Dualité “qualité vs temps” $\rightsquigarrow$ “diversification vs intensification”

Pourquoi et comment intensifier ?

Augmenter l’effort de recherche sur les zones “prometteuses”

- Dépôt de phéromone sur les meilleures solutions
- Choix des composants en fonction des quantités déposées

Pourquoi et comment diversifier ?

Permettre la découverte de nouvelles zones

- Prise de décision stochastique
- MAX-MIN Ant System :
  - borner les traces entre τ_{min} et τ_{max}
  - initialiser la phéromone à τ_{max}

$\rightsquigarrow$ preuve de convergence à l’optimum... en temps infini !

Influence des paramêtres sur l'intensification/diversification

τ_{min}, τ_{max} : bornes min et max de la phéromone

$\rightsquigarrow$ l'intensification augmente quand $\tau_{max} - \tau_{min}$ augmente

nbAnts : nombre de fourmis

$\rightsquigarrow$ la diversification augmente quand *nbAnts* augmente

α : Poids du facteur phéromonal

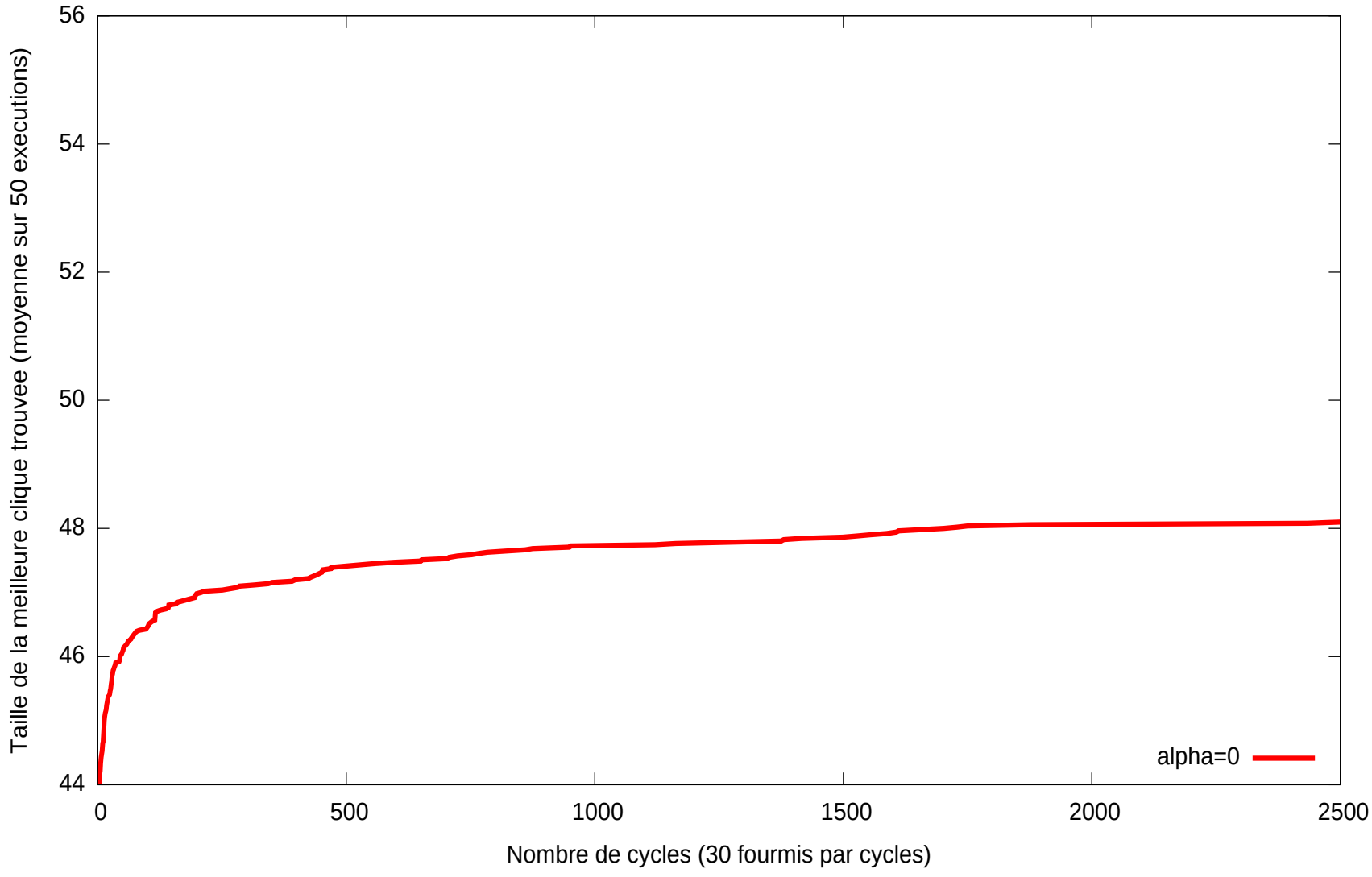
$\rightsquigarrow$ l'intensification augmente quand α augmente

ρ : taux d'évaporation

$\rightsquigarrow$ l'intensification augmente quand ρ augmente

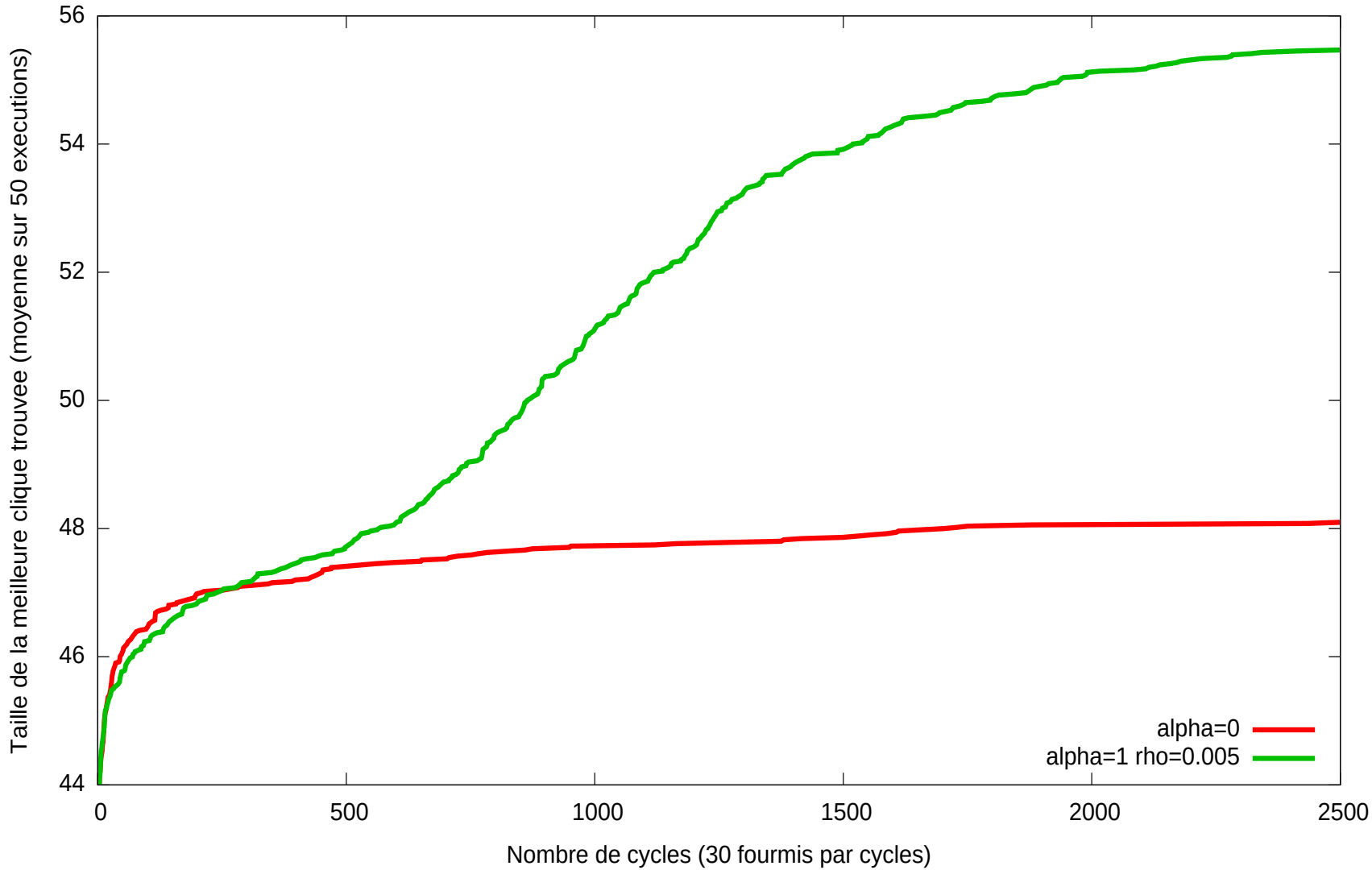
Le meilleur paramétrage dépend du temps disponible !

Illustration sur un problème de recherche de clique



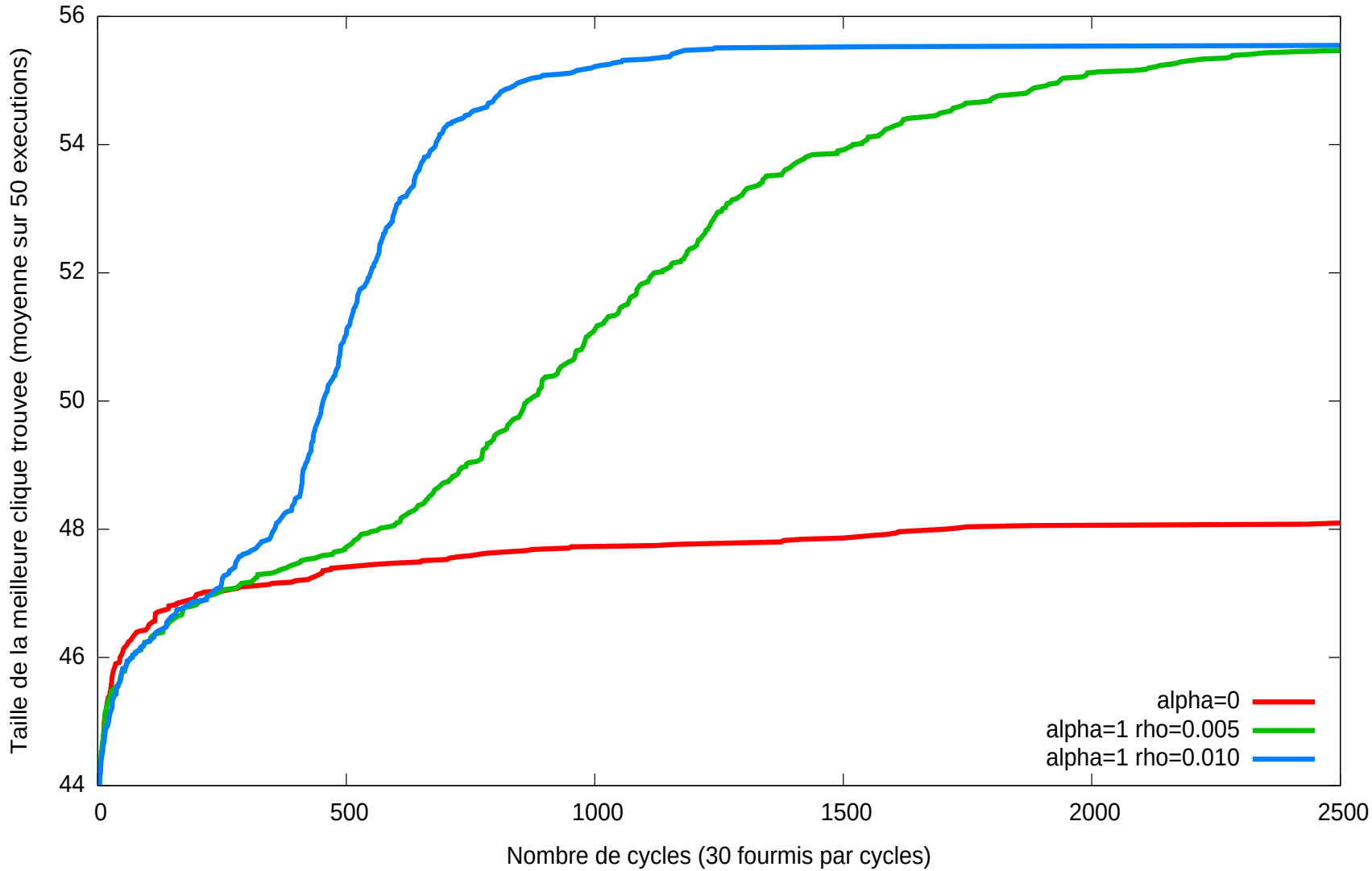
Sans phéromone : $\alpha = 0$

Illustration sur un problème de recherche de clique



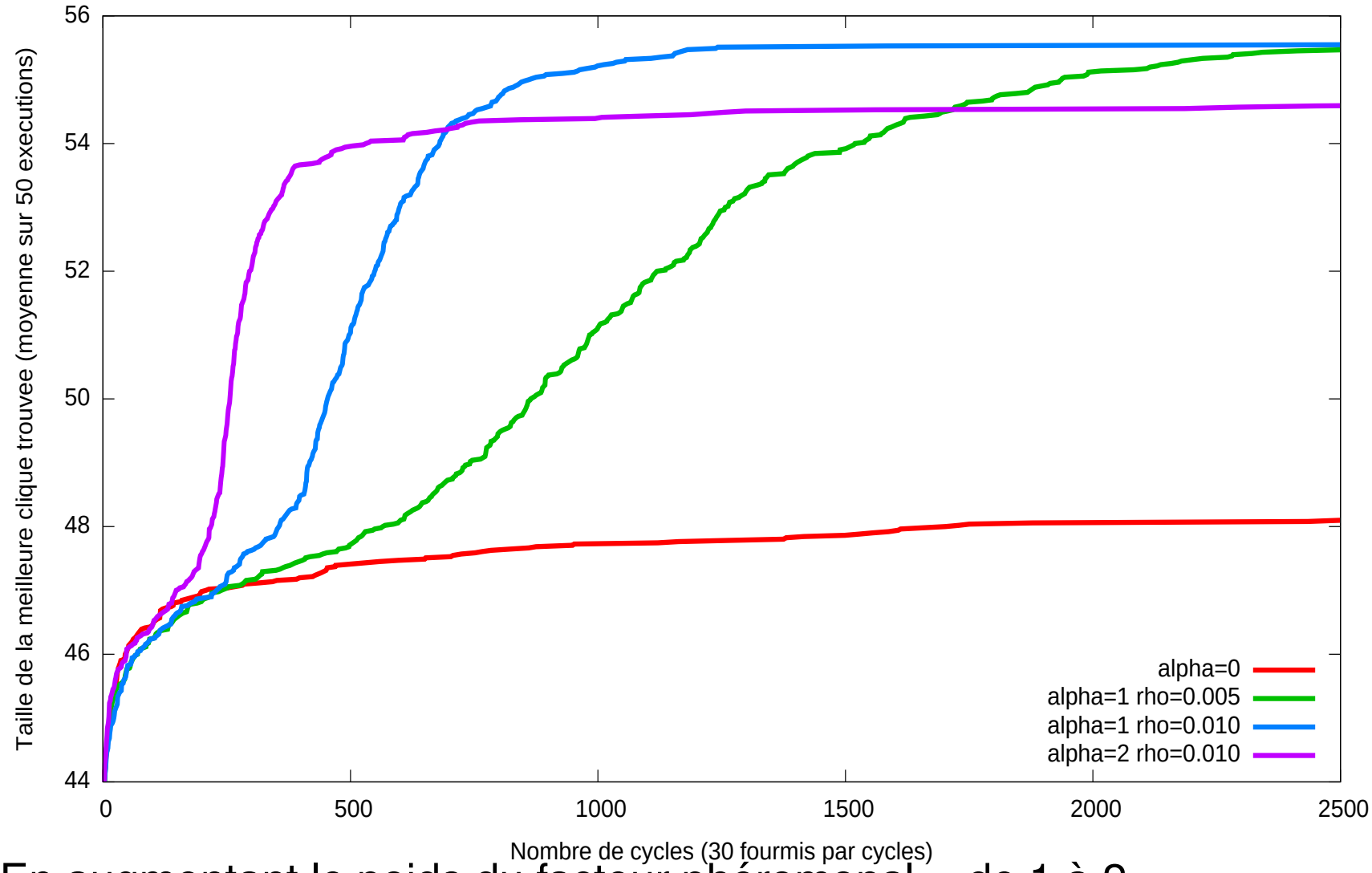
Avec une petite influence de la phéromone : $\alpha = 1$, $\rho = 0.005$

Illustration sur un problème de recherche de clique



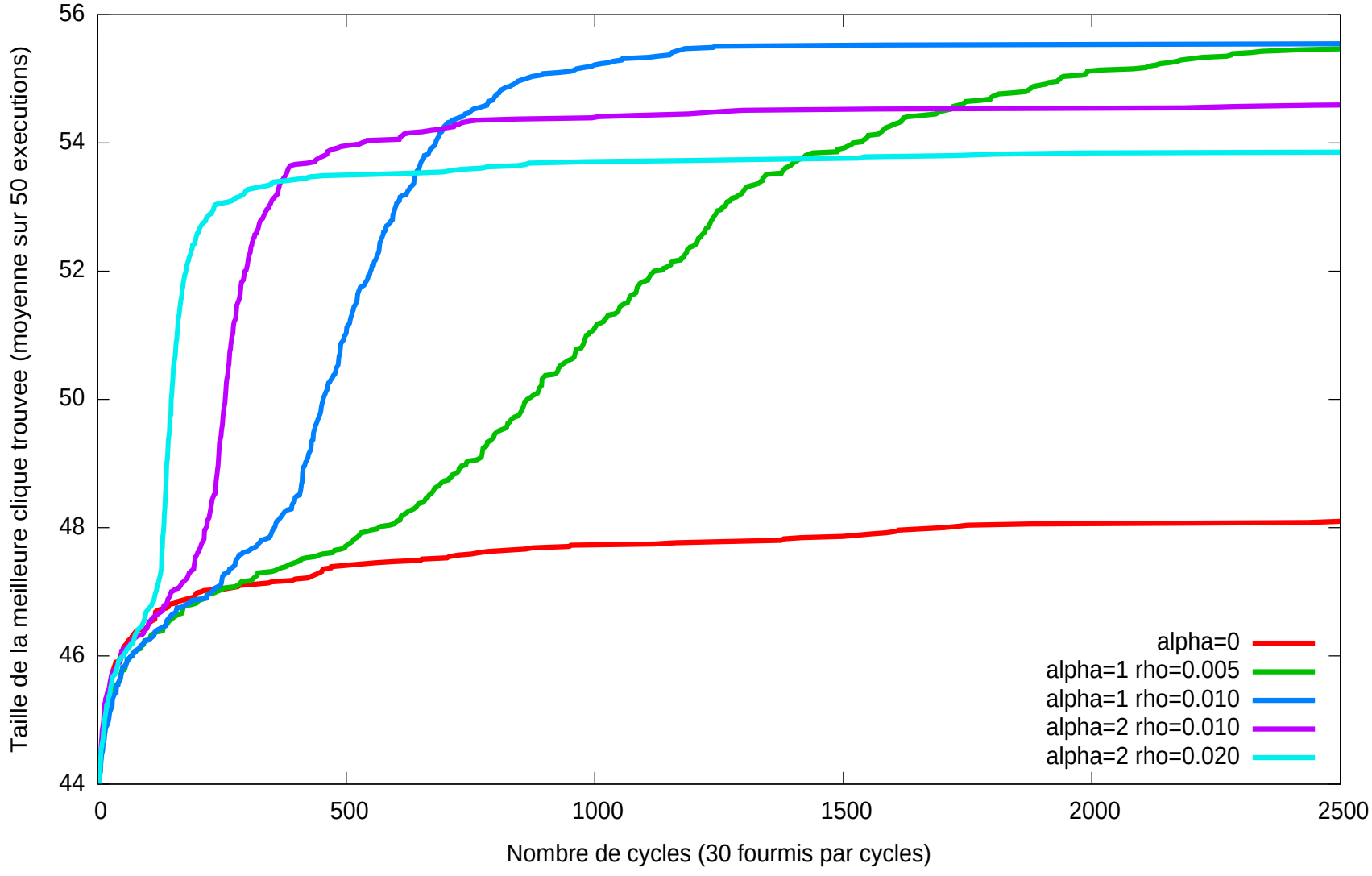
En augmentant l'évaporation ρ de 0.5 % à 1%

Illustration sur un problème de recherche de clique



En augmentant le poids du facteur phéromonal α de 1 à 2

Illustration sur un problème de recherche de clique



En augmentant l'évaporation ρ de 1 % à 2%

Indicateurs d'exploration/intensification

Quantifier l'exploration $\rightsquigarrow$ Taux de ré-échantillonnage

- $\text{Taux} = \frac{\text{nb solutions calculées - nb solutions différentes}}{\text{nb solutions calculées}}$
- Exploration maximale $\Leftarrow 0 \leq \text{Taux} \leq 1 \Rightarrow$ Stagnation

Quantifier l'intensification $\rightsquigarrow$ Taux de similarité

- Taux = similarité moyenne des solutions calculées S
  $\rightsquigarrow$ moyenne des similarités des paires de solutions de S
  $\rightsquigarrow$ similarité de 2 solutions = taux de composants communs
- Augmentation du Taux $\rightsquigarrow$ Intensification

Calcul en temps (quasi) constant de ces deux indicateurs !!!

Indicateurs d'exploration/intensification

Quantifier l'exploration $\rightsquigarrow$ Taux de ré-échantillonnage

- $\text{Taux} = \frac{\text{nb solutions calculées - nb solutions différentes}}{\text{nb solutions calculées}}$
- Exploration maximale $\Leftarrow 0 \leq \text{Taux} \leq 1 \Rightarrow$ Stagnation

Quantifier l'intensification $\rightsquigarrow$ Taux de similarité

- Taux = similarité moyenne des solutions calculées S
  $\rightsquigarrow$ moyenne des similarités des paires de solutions de S
  $\rightsquigarrow$ similarité de 2 solutions = taux de composants communs
- Augmentation du Taux $\rightsquigarrow$ Intensification

Calcul en temps (quasi) constant de ces deux indicateurs !!!

Indicateurs d'exploration/intensification

Quantifier l'exploration $\rightsquigarrow$ Taux de ré-échantillonnage

- Taux $= \frac{\text{nb solutions calculées - nb solutions différentes}}{\text{nb solutions calculées}}$
- Exploration maximale $\Leftarrow 0 \leq$ Taux $\leq 1 \Rightarrow$ Stagnation

Quantifier l'intensification $\rightsquigarrow$ Taux de similarité

- Taux = similarité moyenne des solutions calculées S
  $\rightsquigarrow$ moyenne des similarités des paires de solutions de S
  $\rightsquigarrow$ similarité de 2 solutions = taux de composants communs
- Augmentation du Taux $\rightsquigarrow$ Intensification

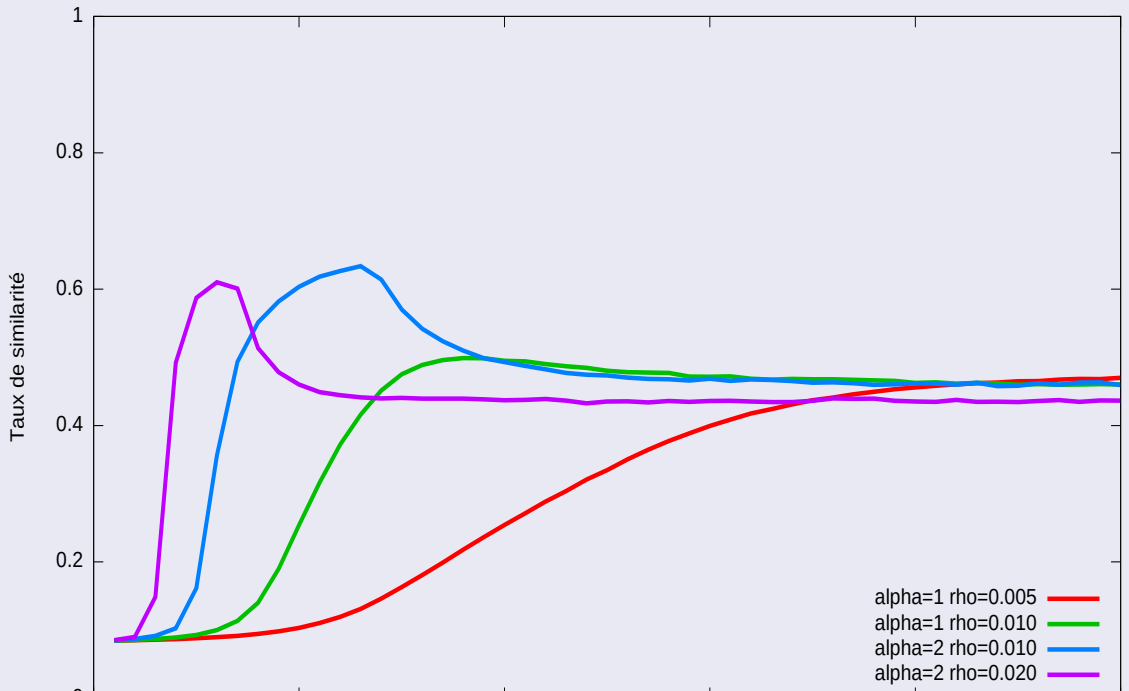
Calcul en temps (quasi) constant de ces deux indicateurs !!!

Indicateurs d'exploration/intensification : Exemple

Taux de ré-échantillonnage

Nombre de cycles:	500	1000	1500	2000	2500
$\alpha = 1, \rho = 0.01$	0.00	0.00	0.00	0.00	0.00
$\alpha = 2, \rho = 0.01$	0.00	0.04	0.06	0.07	0.07
$\alpha = 2, \rho = 0.02$	0.06	0.10	0.12	0.13	0.13

Taux de similarité

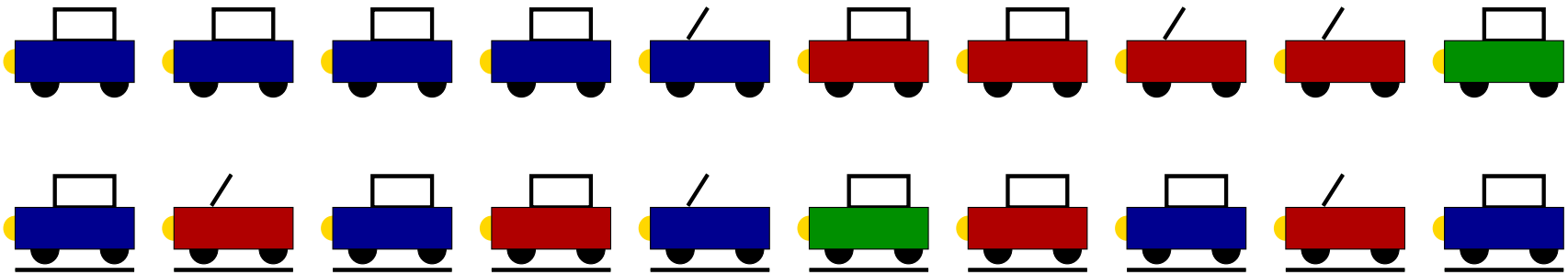


Le problème de l'ordonnancement de voitures (rappel)

Benchmark classique [Dincbas & all 88, Tsang 93, CSP lib]
$\rightsquigarrow$ NP-difficile [Kis 04]

But : Séquencer des voitures sur une chaîne de montage

- Chaque voiture demande un ensemble d'options
- Espacer les voitures demandant une même option



Contraintes à respecter pour le séquencement :

$$Bleu \leq 1/2 \ ; \ Rouge \leq 2/5 \ ; \ Vert \leq 1/5 \ ; \ Cabriolet \leq 1/3$$

ACO pour l'ordonnancement de voitures

- Algorithme glouton aléatoire (rappel)

  $\rightsquigarrow$ heuristique pour identifier les voitures critiques

- ACO 1

  $\rightsquigarrow$ structure phéromonale pour identifier les bonnes sous-séquences de voitures

- ACO 2

  $\rightsquigarrow$ structure phéromonale pour identifier les voitures critiques

- ACO 1+2

  $\rightsquigarrow$ combine les deux structures phéromonales

Algorithme glouton aléatoire

- Partir d'une séquence π vide
- Tant qu'il reste des voitures non séquencées dans π :
  - Soit *cand* l'ensemble des voitures non séquencées
  - restreindre *cand* aux voitures :
    - introduisant le moins de violations
    - demandant des options différentes
  - choisir $c_i \in cand$ selon la probabilité $p(c_i, cand, \pi)$
  - ajouter c_i à la fin de π

Algorithme glouton aléatoire

- Partir d'une séquence π vide
- Tant qu'il reste des voitures non séquencées dans π :
  - Soit *cand* l'ensemble des voitures non séquencées
  - restreindre *cand* aux voitures :
    - introduisant le moins de violations
    - demandant des options différentes
  - choisir $c_i \in cand$ selon la **probabilité** $p(c_i, cand, \pi)$
  - ajouter c_i à la fin de π

$$p(c_i, cand, \pi) = \frac{[\eta(c_i, \pi)]^\beta}{\sum_{c_k \in cand} [\eta(c_k, \pi)]^\beta}$$

où

- $\eta(c_i, \pi)$ = fonction heuristique
  $\rightsquigarrow$ somme des taux d'utilisation des options demandées par c_i
- β = paramètre contrôlant l'influence de l'heuristique

Algorithme ACO pour “apprendre les séquences”

Structure phéromonale

Phéromone sur les couples de voitures

$\rightsquigarrow \tau_1(c_i, c_j)$ = qté de phéro. sur (c_i, c_j)

$\rightsquigarrow$ expérience de la colonie / séquencer c_j juste derrière c_i

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter la voiture c_j à la fin de la séquence π

$$p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} . [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in cand} [\tau_\pi(c_k)]^{\alpha_1} . [\eta_\pi(c_k)]^{\beta}}$$

Algorithme ACO pour “apprendre les séquences”

Structure phéromonale

Phéromone sur les couples de voitures

$\rightsquigarrow \tau_1(c_i, c_j)$ = qté de phéro. sur (c_i, c_j)

$\rightsquigarrow$ expérience de la colonie / séquencer c_j juste derrière c_i

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter la voiture c_j à la fin de la séquence π

$$p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} . [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in cand} [\tau_\pi(c_k)]^{\alpha_1} . [\eta_\pi(c_k)]^{\beta}}$$

Algorithme ACO pour “apprendre les séquences”

Structure phéromonale

Phéromone sur les couples de voitures
$\rightsquigarrow \tau_1(c_i, c_j)$ = qté de phéro. sur (c_i, c_j)
$\rightsquigarrow$ expérience de la colonie / séquencer c_j juste derrière c_i

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter la voiture c_j à la fin de la séquence π

$$p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} . [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in cand} [\tau_\pi(c_k)]^{\alpha_1} . [\eta_\pi(c_k)]^{\beta}}$$

$\tau_\pi(c_j)$ = facteur phéromonal :
$\rightsquigarrow$ si la dernière voiture de π est c_i alors $\tau_\pi(c_j) = \tau_1(c_i, c_j)$
$\rightsquigarrow$ pour choisir la première voiture d'une séquence, $\tau_\pi(c_j) = 1$

Algorithme ACO pour “apprendre les séquences”

Structure phéromonale

Phéromone sur les couples de voitures
$\rightsquigarrow \tau_1(c_i, c_j)$ = qté de phéro. sur (c_i, c_j)
$\rightsquigarrow$ expérience de la colonie / séquencer c_j juste derrière c_i

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter la voiture c_j à la fin de la séquence π

$$p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} . [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in cand} [\tau_\pi(c_k)]^{\alpha_1} . [\eta_\pi(c_k)]^{\beta}}$$

$\eta_\pi(c_j)$ = heuristique locale qui évalue la “difficulté” de c_j
$\rightsquigarrow$ somme des taux d'utilisation des options demandées par c_j

Algorithme ACO pour “apprendre les séquences”

Structure phéromonale

Phéromone sur les couples de voitures
$\rightsquigarrow \tau_1(c_i, c_j)$ = qté de phéro. sur (c_i, c_j)
$\rightsquigarrow$ expérience de la colonie / séquencer c_j juste derrière c_i

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter la voiture c_j à la fin de la séquence π

$$p(c_j) = \frac{[\tau_\pi(c_j)]^{\alpha_1} . [\eta_\pi(c_j)]^{\beta}}{\sum_{c_k \in cand} [\tau_\pi(c_k)]^{\alpha_1} . [\eta_\pi(c_k)]^{\beta}}$$

A la fin de chaque cycle, récompense des meilleures séquences

Ajout de phéromone sur les arcs des meilleures séquences du cycle
Quantité ajoutée $= 1/$nombre de contraintes violées

Algorithme ACO pour “apprendre les voitures critiques”

Structure phéromonale

Traces phéromonales associées aux voitures (regroupées en classes) :

$\rightsquigarrow \tau_2(cc)$ = quantité de phéromone associée à la classe cc

$\rightsquigarrow$ expérience de la colonie concernant la difficulté de cc

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter une voiture de classe cc_j à la fin de la séquence

$$p(cc_j) = \frac{[\tau_2(cc_j)]^{\alpha_2}}{\sum_{cc_k \in cand}[\tau(cc_k)]^{\alpha_2}}$$

Mise-à-jour de la phéromone

- Au cours de la construction :
  $\rightsquigarrow$ ajout de phéromone sur les classes violant des contraintes
- A la fin de la construction :
  $\rightsquigarrow$ évaporation

Algorithme ACO pour “apprendre les voitures critiques”

Structure phéromonale

Traces phéromonales associées aux voitures (regroupées en classes) :

$\rightsquigarrow \tau_2(cc)$ = quantité de phéromone associée à la classe cc

$\rightsquigarrow$ expérience de la colonie concernant la difficulté de cc

A chaque cycle, chaque fourmi construit une séquence

Probabilité d’ajouter une voiture de classe cc_j à la fin de la séquence

$$p(cc_j) = \frac{[\tau_2(cc_j)]^{\alpha_2}}{\sum_{cc_k \in cand}[\tau(cc_k)]^{\alpha_2}}$$

Mise-à-jour de la phéromone

- Au cours de la construction :
  $\rightsquigarrow$ ajout de phéromone sur les classes violant des contraintes
- A la fin de la construction :
  $\rightsquigarrow$ évaporation

Algorithme ACO pour “apprendre les voitures critiques”

Structure phéromonale

Traces phéromonales associées aux voitures (regroupées en classes) :
$\rightsquigarrow \tau_2(cc)$ = quantité de phéromone associée à la classe cc
$\rightsquigarrow$ expérience de la colonie concernant la difficulté de cc

A chaque cycle, chaque fourmi construit une séquence

Probabilité d’ajouter une voiture de classe cc_j à la fin de la séquence

$$p(cc_j) = \frac{[\tau_2(cc_j)]^{\alpha 2}}{\sum_{cc_k \in cand}[\tau(cc_k)]^{\alpha 2}}$$

Mise-à-jour de la phéromone

- Au cours de la construction :
  $\rightsquigarrow$ ajout de phéromone sur les classes violant des contraintes
- A la fin de la construction :
  $\rightsquigarrow$ évaporation

Double algorithme ACO

Combiner les deux structures phéromonales

- Pour “apprendre les bonnes séquences” : $\forall$ voitures c_i et c_j
  $\tau_1(c_i, c_j)$ = expérience de la colonie / séquencer c_j après c_i
- Pour “apprendre les voitures critiques” : $\forall$ classe de voitures cc
  $\tau_2(cc)$ = expérience de la colonie / séquencer les voitures de cc

A chaque cycle, chaque fourmi construit une séquence

Probabilité d’ajouter la voiture c_j à la fin de la séquence

$$p(c_j) = \frac{[\tau_1(c_i,c_j)]^{\alpha_1}.[\tau_2(classOf(c_j))]^{\alpha_2}}{\sum_{c_k \in cand}[\tau_1(c_i,c_k)]^{\alpha_1}.[\tau_2(classOf(c_k))]^{\alpha_2}}$$

Mise-à-jour des traces de phéromone

$\rightsquigarrow$ En cours de construction d’une séquence : dépôts sur $\tau_2(cc)$
$\rightsquigarrow$ Fin de construction d’une séquence : évaporation de $\tau_2(cc)$
$\rightsquigarrow$ Fin d’un cycle : évaporation + dépots sur $\tau_1(c_i, c_j)$

Double algorithme ACO

Combiner les deux structures phéromonales

- Pour “apprendre les bonnes séquences” : $\forall$ voitures c_i et c_j
  $\tau_1(c_i, c_j)$ = expérience de la colonie / séquencer c_j après c_i
- Pour “apprendre les voitures critiques” : $\forall$ classe de voitures cc
  $\tau_2(cc)$ = expérience de la colonie / séquencer les voitures de cc

A chaque cycle, chaque fourmi construit une séquence

Probabilité d’ajouter la voiture c_j à la fin de la séquence

$$p(c_j) = \frac{[\tau_1(c_i,c_j)]^{\alpha 1}.[\tau_2(classOf(c_j))]^{\alpha 2}}{\sum_{c_k \in cand}[\tau_1(c_i,c_k)]^{\alpha 1}.[\tau_2(classOf(c_k))]^{\alpha 2}}$$

Mise-à-jour des traces de phéromone

- ⇝ En cours de construction d’une séquence : dépôts sur $\tau_2(cc)$
- ⇝ Fin de construction d’une séquence : évaporation de $\tau_2(cc)$
- ⇝ Fin d’un cycle : évaporation + dépots sur $\tau_1(c_i, c_j)$

Double algorithme ACO

Combiner les deux structures phéromonales

- Pour "apprendre les bonnes séquences" : $\forall$ voitures c_i et c_j
  $\tau_1(c_i, c_j)$ = expérience de la colonie / séquencer c_j après c_i
- Pour "apprendre les voitures critiques" : $\forall$ classe de voitures cc
  $\tau_2(cc)$ = expérience de la colonie / séquencer les voitures de cc

A chaque cycle, chaque fourmi construit une séquence

Probabilité d'ajouter la voiture c_j à la fin de la séquence

$$p(c_j) = \frac{[\tau_1(c_i,c_j)]^{\alpha 1}.[\tau_2(classOf(c_j))]^{\alpha 2}}{\sum_{c_k \in cand}[\tau_1(c_i,c_k)]^{\alpha 1}.[\tau_2(classOf(c_k))]^{\alpha 2}}$$

Mise-à-jour des traces de phéromone

$\rightsquigarrow$ En cours de construction d'une séquence : dépôts sur $\tau_2(cc)$
$\rightsquigarrow$ Fin de construction d'une séquence : évaporation de $\tau_2(cc)$
$\rightsquigarrow$ Fin d'un cycle : évaporation + dépots sur $\tau_1(c_i, c_j)$

Résultats expérimentaux

Paramétrage et conditions expérimentales

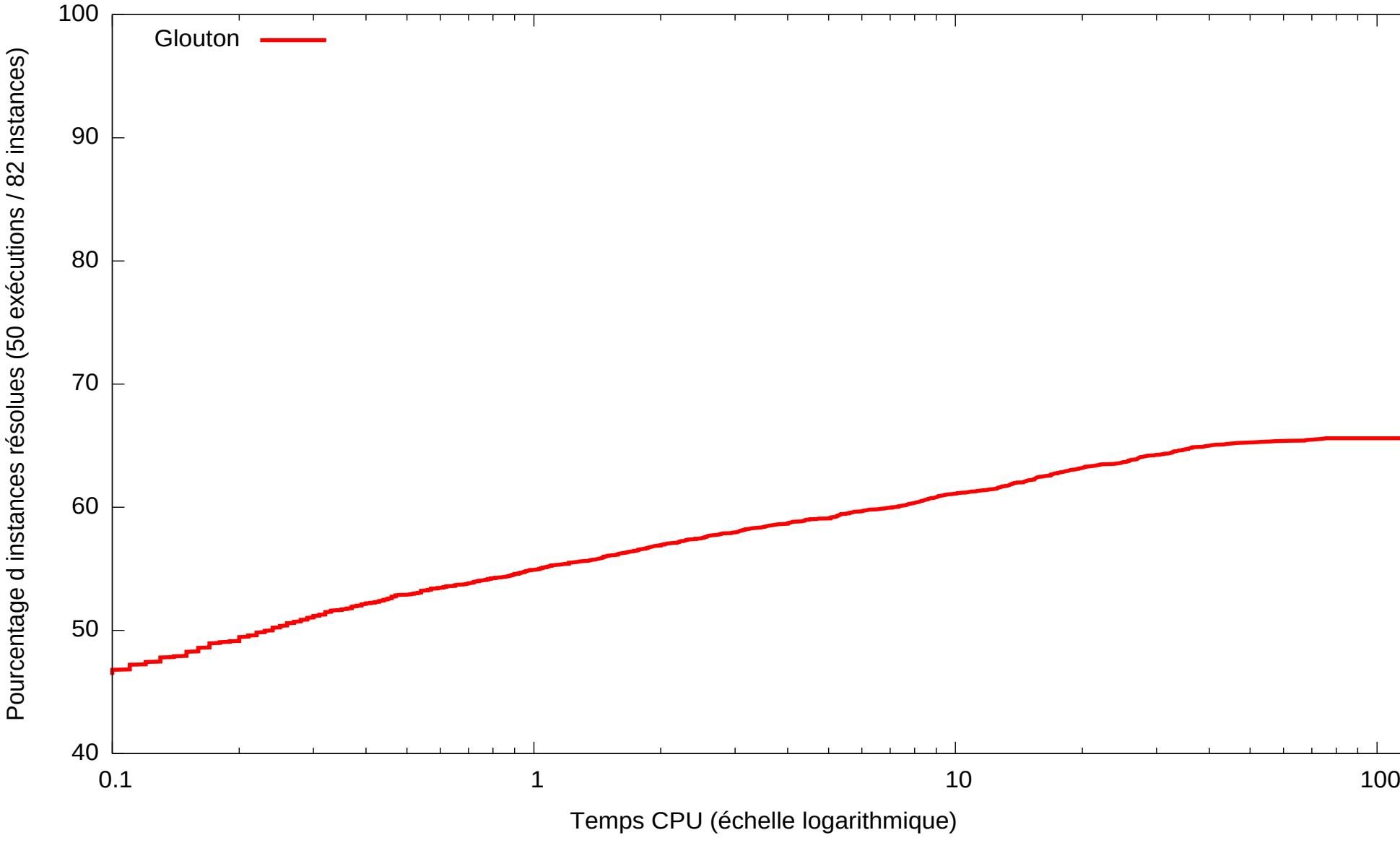
Algo	Heur. η β	Phéromone 1 α_1	ρ_1	τ_{min_1}	τ_{max_1}	Phéromone 2 α_2	ρ_2	τ_{min_2}
Greedy(η)	6	-	-	-	-	-	-	-
ACO(τ_1, η)	6	2	1%	0.01	4	-	-	-
ACO(τ_2)	-	-	-	-	-	6	3%	1
ACO(τ_1, τ_2)	-	2	1%	0.01	4	6	3%	1

5000 cycles, 30 fourmis $\rightsquigarrow$ 150000 constructions gloutonnes
50 exécutions par instance sur un Pentium 4 à 2GHz

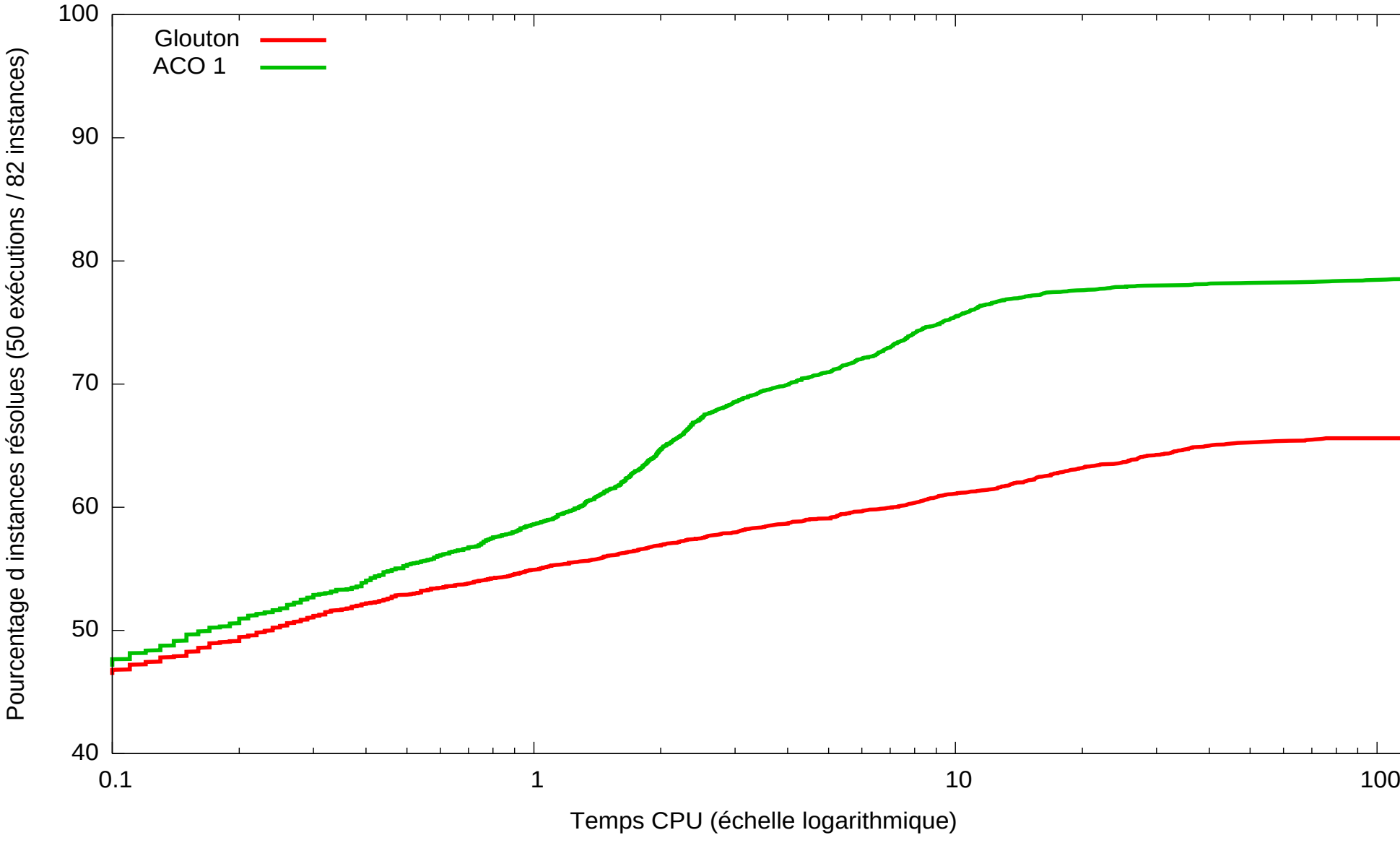
Jeux d'essais

- Instances de CSP Lib toutes résolues ($<$ 20 constructions)
- Instances de [Perron & Shaw 04]
  - Toutes satisfiables
  - 32, 21 et 29 instances à 100, 300 et 500 voitures
  - 20 classes et 8 options

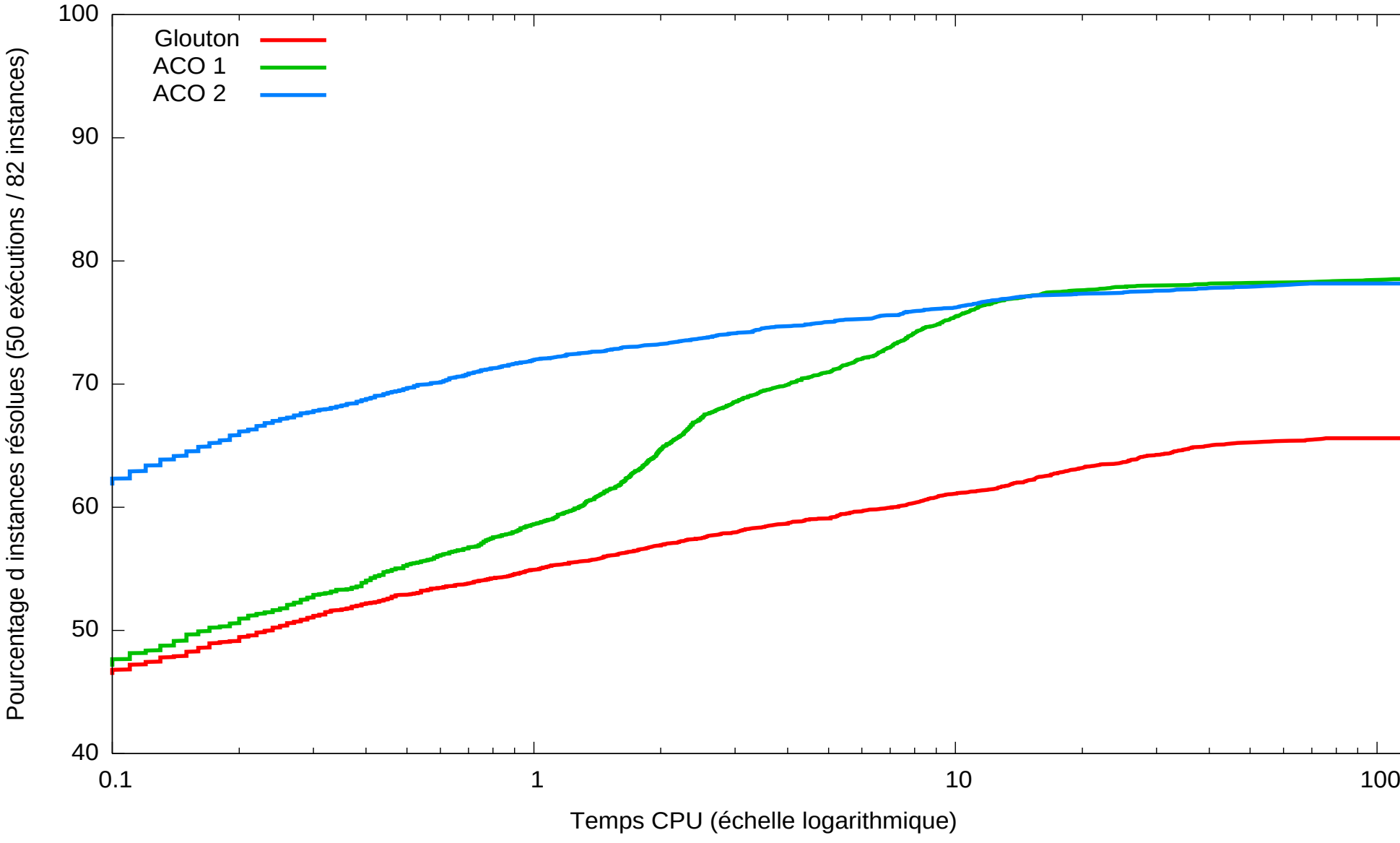
Comparaison : % succès / temps CPU



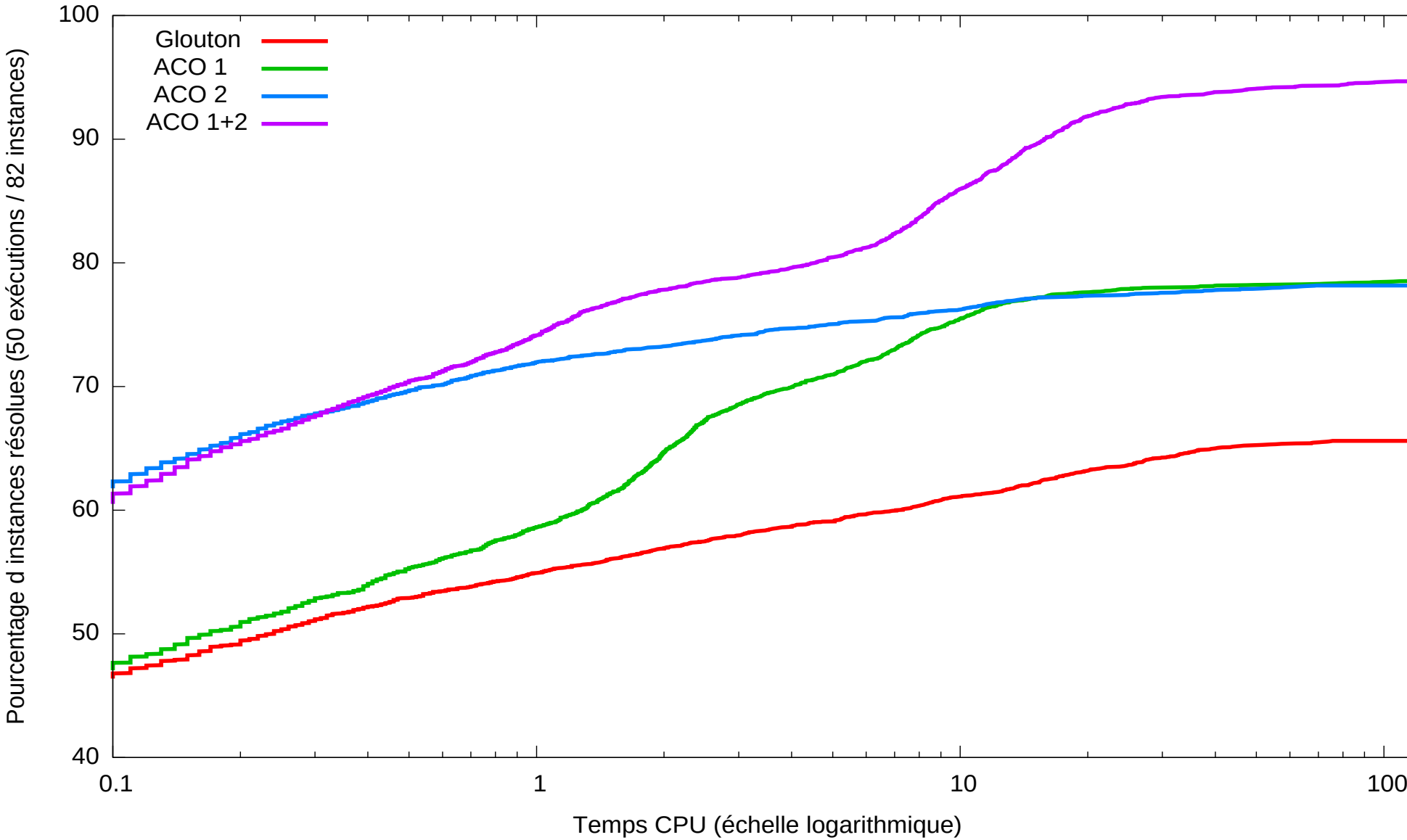
Comparaison : % succès / temps CPU



Comparaison : % succès / temps CPU



Comparaison : % succès / temps CPU

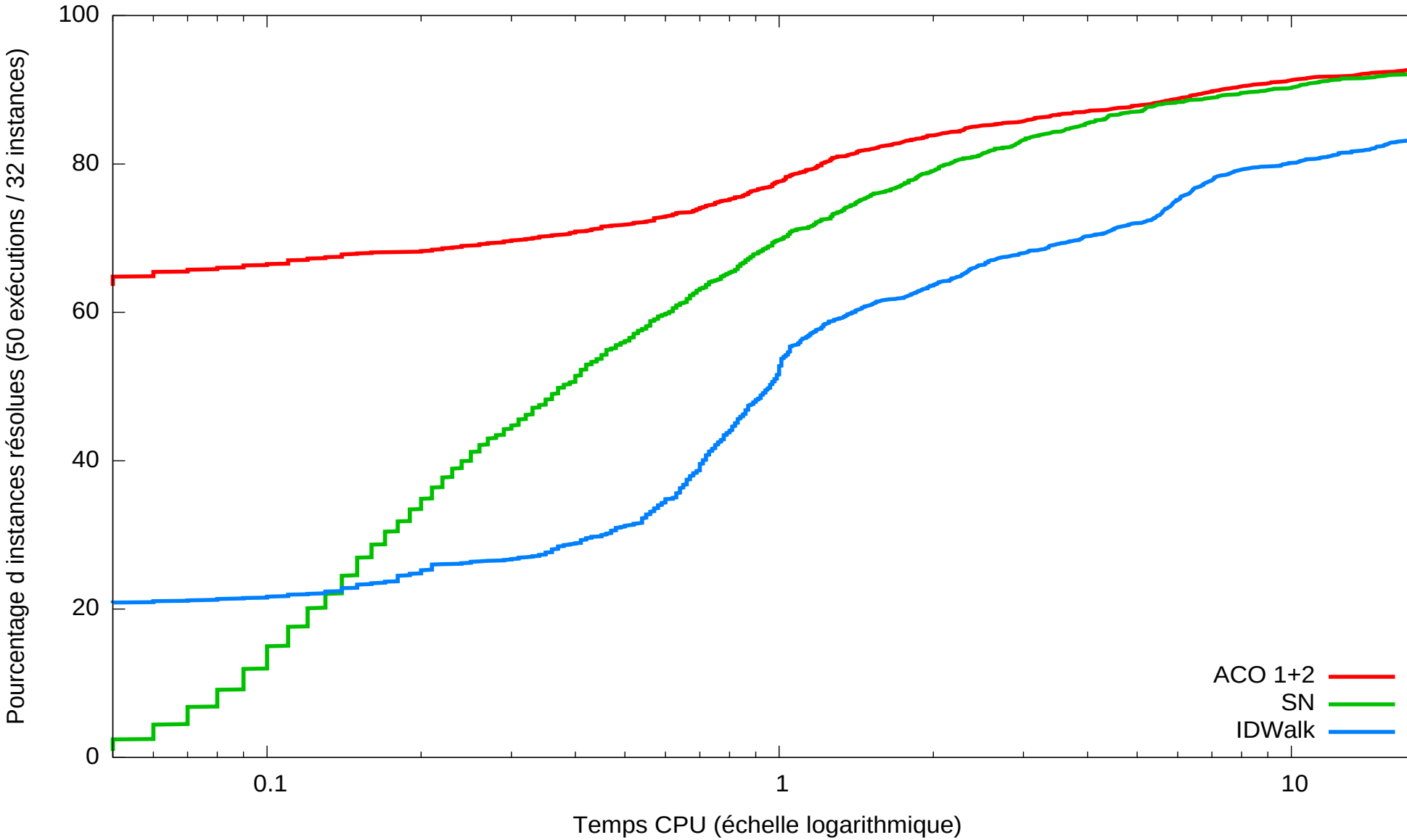


Comparaison avec d'autres approches

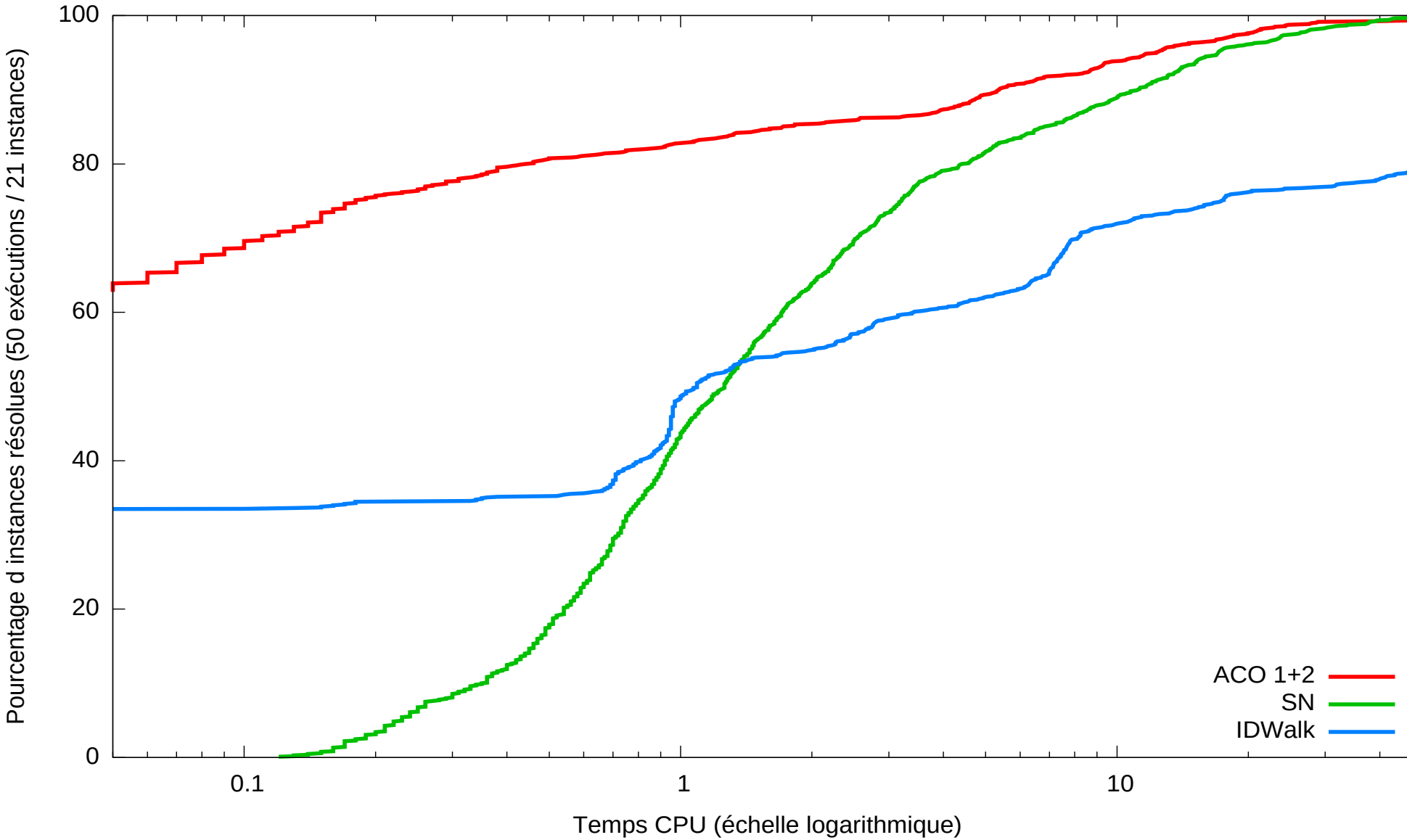
Approches considérées

- SN = Vainqueur du Challenge ROADEF 2005 [Estellon et al. 05]
  Exploration de l'espace des combinaisons par recherche locale :
  - Point de départ généré par un algorithme glouton
  - Voisinage = échange / miroir / insertion / permutation
  - Premier voisin non détériorant
- ID Walk [Neveu et al. 04]
  Exploration de l'espace des combinaisons par recherche locale :
  - Point de départ généré par un algorithme glouton
  - Voisinage = échange
  - *Max* voisins explorés au maximum à chaque mouvement
    $\rightsquigarrow$ Premier voisin non détériorant...
    ... ou le moins mauvais parmi les *max* explorés
    $\rightsquigarrow$ réglage automatique de la valeur de *Max*

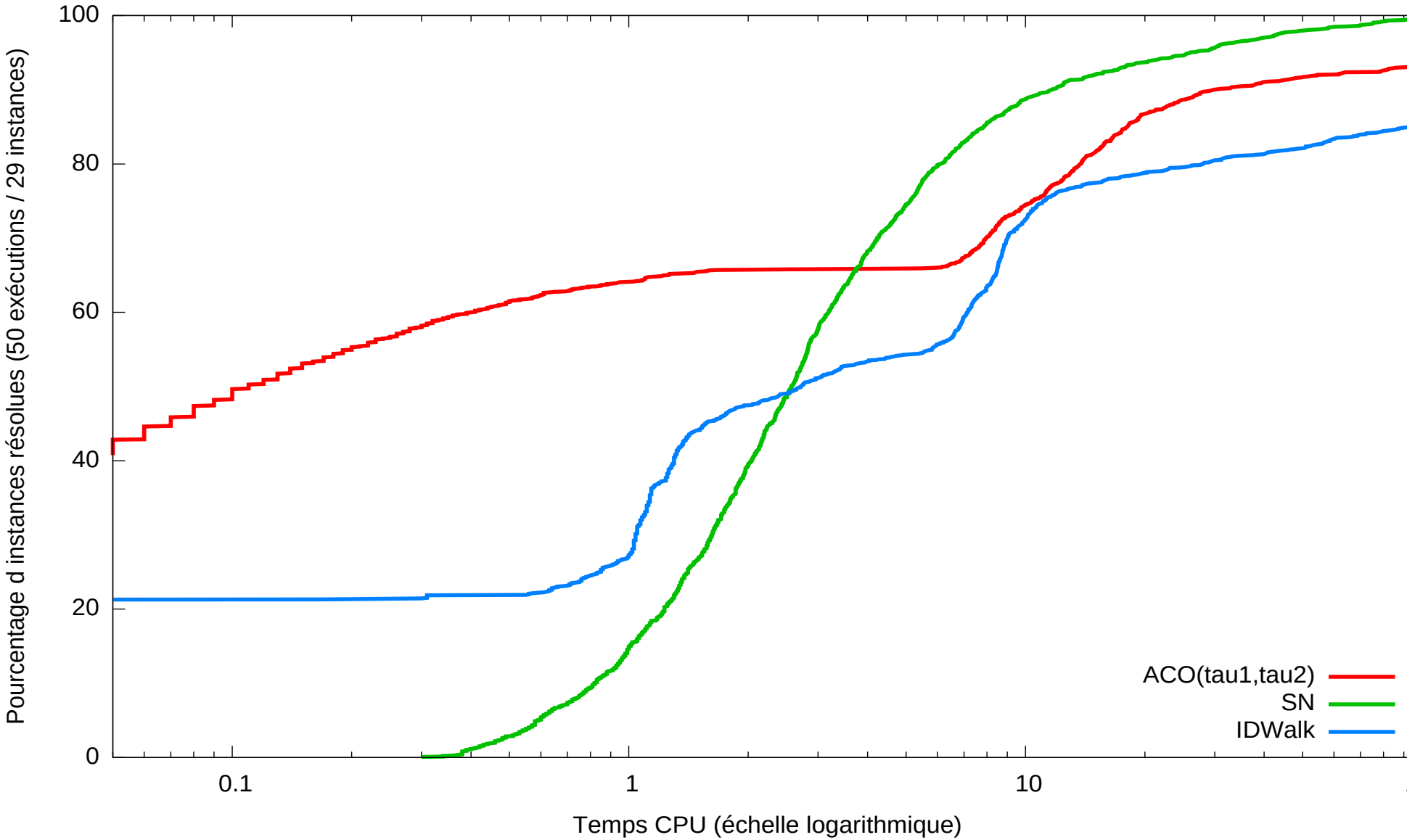
Comparaison : 32 instances à 100 voitures



Comparaison : 21 instances à 300 voitures



Comparaison : 29 instances à 500 voitures



Problèmes de recherche de sous-ensembles

Définition

- Un SS-problème est défini par (S, S_C, f) tel que
  - S = ensemble d'objets candidats
  - $S_C \subseteq \mathcal{P}(S)$ = ensemble des sous-ensembles consistants
  - $f : S_c \rightarrow \mathbb{R}$ = fonction objectif
- But du jeu : trouver $S^* \in S_C$ tel que $f(S^*)$ soit maximal

Exemples

- Recherche de cliques maximum
- Problèmes de sac-à-dos multidimensionnels
- Problèmes de recouvrement d'ensembles
- Problèmes de satisfaction de contraintes :
- ...

Application de ACO à la recherche de sous-ensembles

Stratégie "Vertex" : Apprendre les bons objets

Phéromone associée aux objets : $\tau(i) \rightsquigarrow$ intéret de choisir i

- Sac-à-dos [Leguizamon & Michalewicz 99]
- Recouvrement d'ensembles [Hadji & al 00]
- K-arbres de poids minimum [Blum 02]
- Satisfaction de contraintes [Solnon & Bridge 06]
- Cliques maximum [Solnon & Fenet 06]

Stratégie "Clique" : Apprendre les bonnes paires d'objets

Phéromone associée aux paires : $\tau(i,j) \rightsquigarrow$ intéret de choisir i avec j

- Satisfaction de contraintes [Solnon 02]
- K-arbres de poids minimum [Blum 02]
- Cliques maximum [Fenet & Solnon 03]
- Sac-à-dos [Alaya, Solnon & Ghédira 04]
- Appariement de graphes [Sammoud, Solnon & Ghédira 05]

Application de ACO à la recherche de sous-ensembles

Stratégie "Vertex" : Apprendre les bons objets

Phéromone associée aux objets : $\tau(i) \rightsquigarrow$ intéret de choisir i

- Sac-à-dos [Leguizamon & Michalewicz 99]
- Recouvrement d'ensembles [Hadji & al 00]
- K-arbres de poids minimum [Blum 02]
- Satisfaction de contraintes [Solnon & Bridge 06]
- Cliques maximum [Solnon & Fenet 06]

Stratégie "Clique" : Apprendre les bonnes paires d'objets

Phéromone associée aux paires : $\tau(i,j) \rightsquigarrow$ intéret de choisir i avec j

- Satisfaction de contraintes [Solnon 02]
- K-arbres de poids minimum [Blum 02]
- Cliques maximum [Fenet & Solnon 03]
- Sac-à-dos [Alaya, Solnon & Ghédira 04]
- Appariement de graphes [Sammoud, Solnon & Ghédira 05]

Algorithme ACO pour la recherche de sous-ensembles

Un algorithme générique paramétré par :

$\rightsquigarrow$ un problème à résoudre (S, S_C, f)
$\rightsquigarrow$ une stratégie phéromonale $\Phi \in \{Vertex, Clique\}$

A chaque cycle, chaque fourmi construit un sous-ensemble

Probabilité d'ajouter un objet $o_i \in Candidats$ au sous-ensemble S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

Algorithme ACO pour la recherche de sous-ensembles

Un algorithme générique paramétré par :

$\rightsquigarrow$ un problème à résoudre (S, S_C, f)
$\rightsquigarrow$ une stratégie phéromonale $\Phi \in \{Vertex, Clique\}$

A chaque cycle, chaque fourmi construit un sous-ensemble

Probabilité d'ajouter un objet $o_i \in Candidats$ au sous-ensemble S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

Algorithme ACO pour la recherche de sous-ensembles

Un algorithme générique paramétré par :

$\rightsquigarrow$ un problème à résoudre (S, S_C, f)
$\rightsquigarrow$ une stratégie phéromonale $\Phi \in \{Vertex, Clique\}$

A chaque cycle, chaque fourmi construit un sous-ensemble

Probabilité d'ajouter un objet $o_i \in Candidats$ au sous-ensemble S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

$\tau_{factor}(o_i, S_k) \rightsquigarrow$ dépend de la stratégie phéromonale Φ

Si $\Phi = Vertex$:
$\tau_{factor}(o_i, S_k) = \tau(o_i)$

Si $\Phi = Clique$:
$\tau_{factor}(o_i, S_k) = \sum_{o_j \in S_k} \tau(o_j, o_i)$

Algorithme ACO pour la recherche de sous-ensembles

Un algorithme générique paramétré par :

$\rightsquigarrow$ un problème à résoudre (S, S_C, f)
$\rightsquigarrow$ une stratégie phéromonale $\Phi \in \{Vertex, Clique\}$

A chaque cycle, chaque fourmi construit un sous-ensemble

Probabilité d'ajouter un objet $o_i \in Candidats$ au sous-ensemble S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

$\eta_{factor}(o_i, S_k) \rightsquigarrow$ dépend du problème à résoudre

Algorithme ACO pour la recherche de sous-ensembles

Un algorithme générique paramétré par :

$\rightsquigarrow$ un problème à résoudre (S, S_C, f)
$\rightsquigarrow$ une stratégie phéromonale $\Phi \in \{Vertex, Clique\}$

A chaque cycle, chaque fourmi construit un sous-ensemble

Probabilité d'ajouter un objet $o_i \in Candidats$ au sous-ensemble S_k

$$p(o_i) = \frac{[\tau_{factor}(o_i, S_k)]^{\alpha} . [\eta_{factor}(o_i, S_k)]^{\beta}}{\sum_{o_j \in Candidats} [\tau_{factor}(o_j, S_k)]^{\alpha} . [\eta_{factor}(o_j, S_k)]^{\beta}}$$

A la fin de chaque cycle, récompense des meilleurs sous-ens.

Composants de S_k récompensés $\rightsquigarrow$ dépend de la stratégie Φ

Si Φ = *Vertex* :
dépôt sur les sommets

Si Φ = *Clique* :
dépôt sur les arêtes de la clique

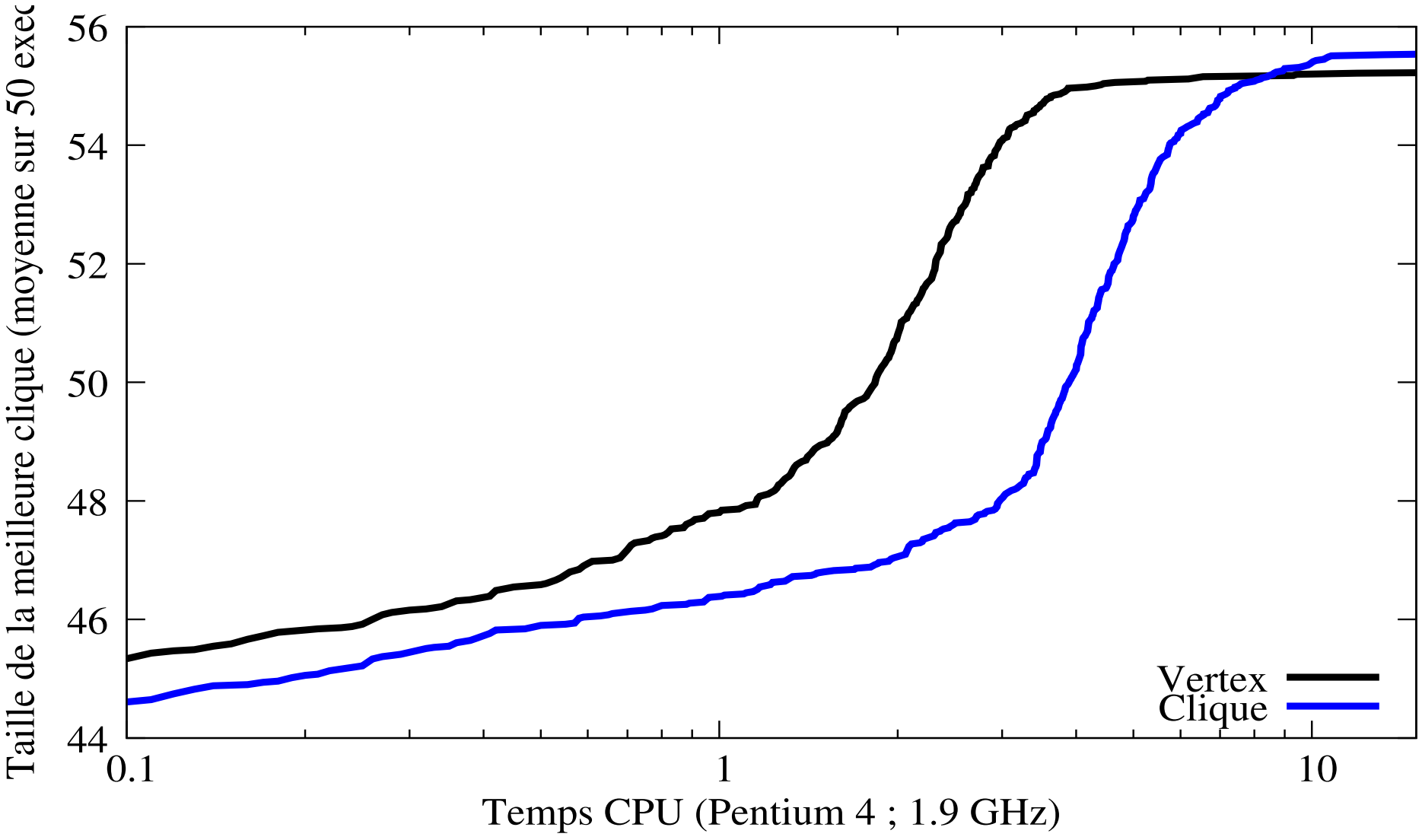
Exemple 1 : Clique maximum

Construction d'une clique C pour un graphe $G = (S, A)$

- $C \leftarrow \emptyset$
- $Cand \leftarrow S$
- tant que $Cand \neq \emptyset$ faire
  - Choisir $s_j \in Cand$ selon la probabilité
    $p(s_j) = \frac{\tau_{factor}(s_j, C)^{\alpha}}{\sum_{s_k \in Cand} \tau_{factor}(s_k, C)^{\alpha}}$
  - $C \leftarrow C \cup \{s_j\}$
  - $Cand \leftarrow Cand \cap \{s_i / (s_i, s_j) \in A\}$

- Pas de facteur heuristique !!!
- $\tau_{factor}(s_j, C)$ dépend de la stratégie phéromonale :
  - Stratégie vertex $\rightsquigarrow \tau_{factor}(s_j, C) = \tau_{s_j}$
  - Stratégie clique $\rightsquigarrow \tau_{factor}(s_j, C) = \sum_{s_k \in C} \tau_{(s_j, s_k)}$

Exemple 1 : Clique maximum



$\alpha = 1; \rho = 1\%; 30$ fourmis

Hybridation ACO / recherche locale

ACO peut être facilement hybridé avec de la recherche locale

- Les fourmis construisent des solutions ≪ initiales ≫ ...

  ... qui sont améliorées par recherche locale

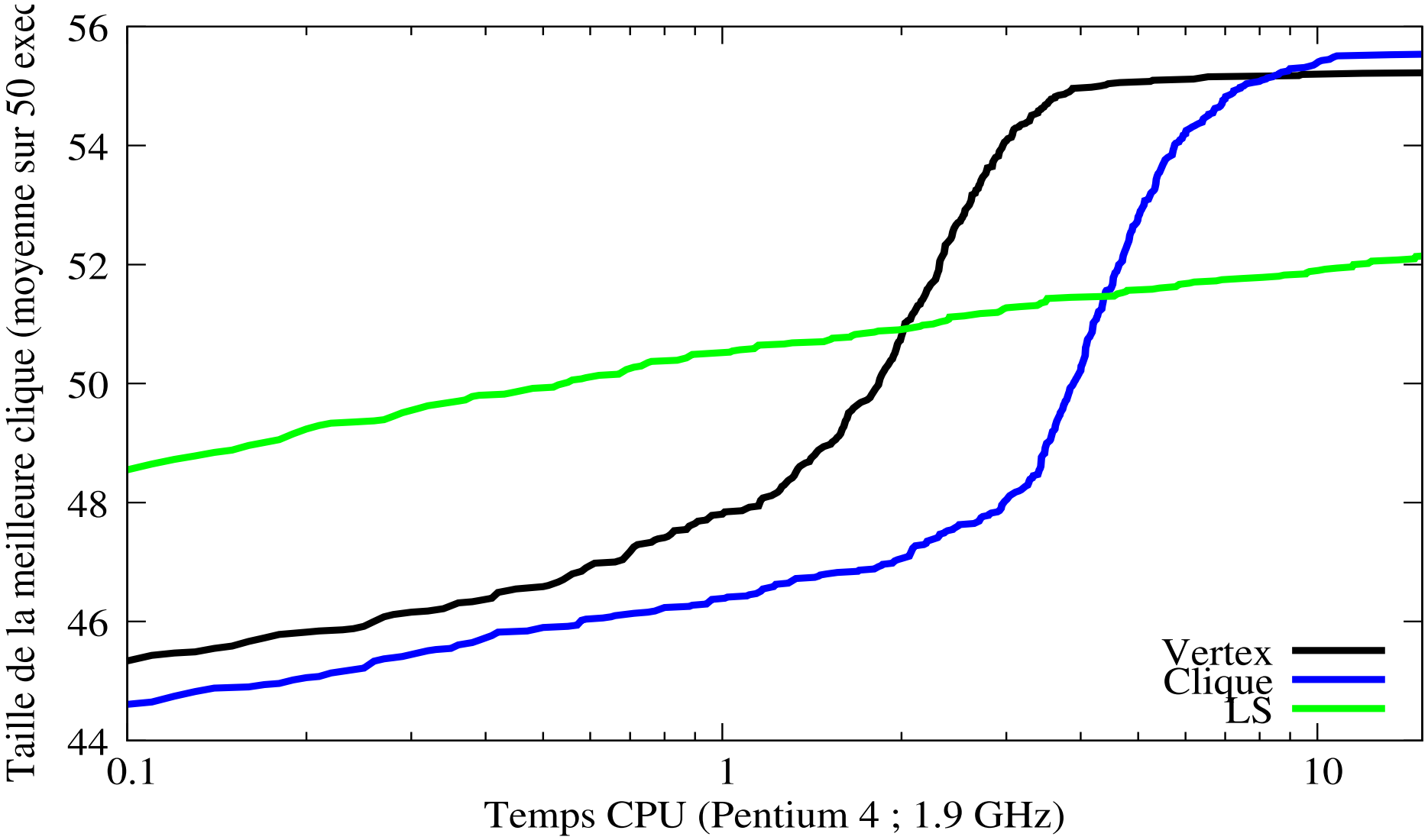
- Choix d'un algorithme de recherche locale

  $\rightsquigarrow$ Trouver un compromis entre temps et qualité

  $\rightsquigarrow$ En général : Recherche locale gloutonne

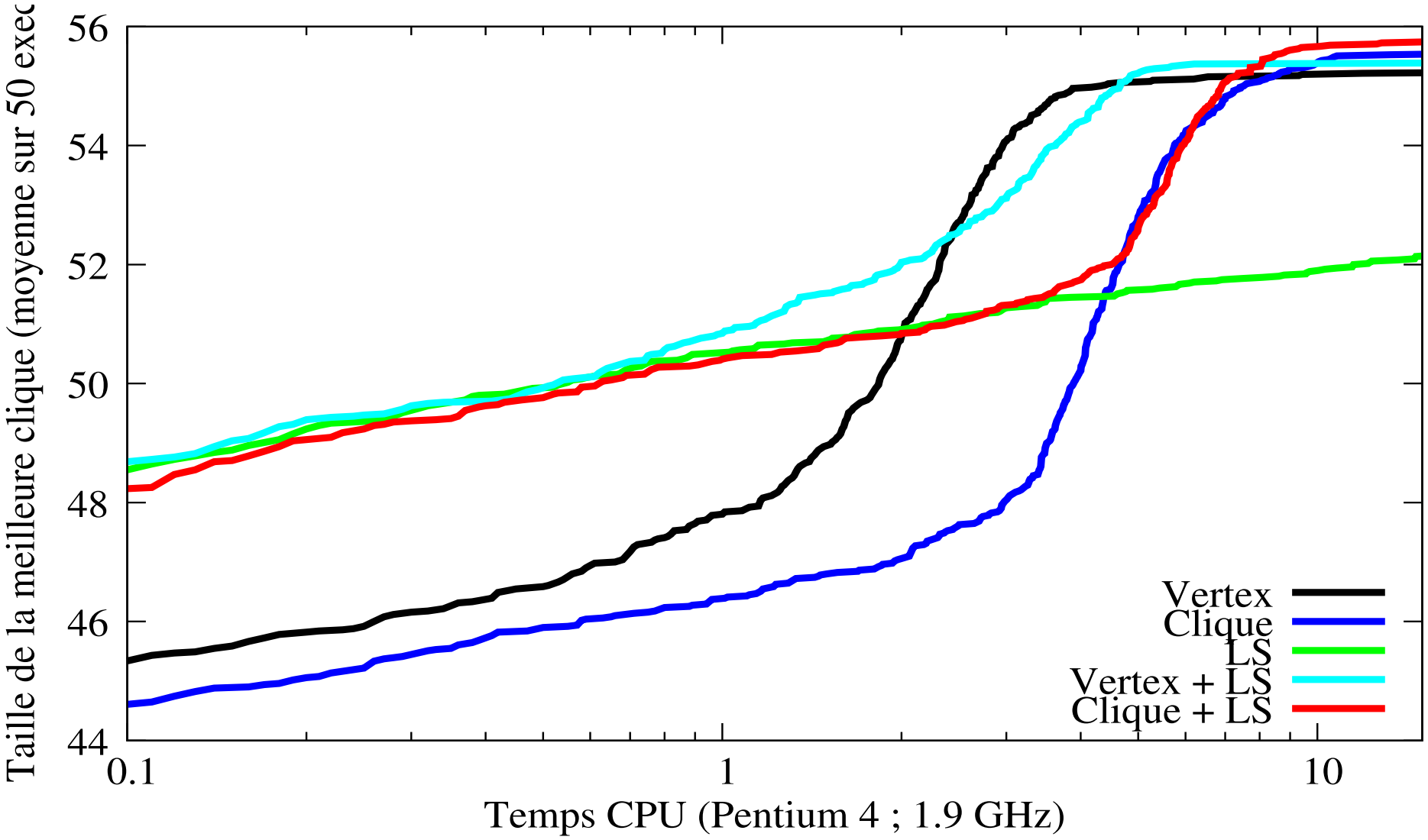
- ACO + recherche locale :
  - ACO $\rightsquigarrow$ calculer de nouveaux points de départ à partir des optima locaux trouvés par la recherche locale
  - Recherche locale $\rightsquigarrow$ amélioration des solutions calculées par ACO

Exemple 1 : Clique maximum



Recherche locale gloutonne / $(2,3)$-échange

Exemple 1 : Clique maximum



Recherche locale gloutonne / $(2,3)$-échange

Exemple 1 : Clique maximum

Taille des cliques trouvées

Graphe	Sans recherche locale				Avec recherche locale			
	Vertex		*Clique*		*Vertex*		*Clique*	
C125.9	**34.0**	**(34)**	**34.0**	**(34)**	**34.0**	**(34)**	**34.0**	**(34)**
C250.9	43.9	**(44)**	**44.0**	**(44)**	**44.0**	**(44)**	**44.0**	**(44)**
C500.9	55.2	(56)	55.6	(57)	55.3	(56)	**55.9**	**(57)**
C1000.9	65.3	(67)	66.0	(67)	65.7	(67)	**66.2**	**(68)**
C2000.9	73.4	(76)	74.1	(76)	**74.5**	(77)	74.3	**(78)**

Nombre de cycles et temps de convergence

Graphe	Sans recherche locale				Avec recherche locale			
	Vertex		*Clique*		*Vertex*		*Clique*	
C125.9	60	0.1	126	0.2	**14**	**0.0**	23	**0.0**
C250.9	359	0.8	473	1.7	**172**	**0.5**	239	1.0
C500.9	722	**3.8**	923	8.9	**477**	4.6	671	8.6
C1000.9	1219	**13.2**	2359	55.0	**832**	23.4	1242	49.8
C2000.9	1770	**41.3**	3268	214.4	**1427**	112.4	2067	238.7

Exemple 2 : CSP

Construction d'une affectation totale A pour un CSP (X, D, C)

$A \leftarrow \emptyset$
$Cand \leftarrow \{\langle X_i, v_i\rangle \mid X_i \in X, v_i \in D(X_i)\}$
tant que $Cand \neq \emptyset$ faire
  Choisir $\langle X_i, v_i\rangle \in Cand$ selon la probabilité

$$p(\langle X_i, v_i\rangle) = \frac{\tau_{factor}(\langle X_i, v_i\rangle, A)^{\alpha} \cdot \eta_{factor}(\langle X_i, v_i\rangle, A)^{\beta}}{\sum_{\langle X_k, v_k\rangle \in Cand} \tau_{factor}(\langle X_k, v_k\rangle, A)^{\alpha} \cdot \eta_{factor}(\langle X_k, v_k\rangle, A)^{\beta}}$$

  $A \leftarrow A \cup \{\langle X_i, v_i\rangle\}$
  enlever de $Cand$ tous les couples $\langle X_i, v_i'\rangle$
  filtrer les domaines (forward checking) / $X_i = v_i$

- Facteur heuristique : "min-dom" + minimiser violations
  $\eta_{factor}(\langle X_i, v_i\rangle, A) = 0$ si $\exists X_j, |D(X_j)| < |D(X_i)|$
  $\eta_{factor}(\langle X_i, v_i\rangle, A) = 1/(1 + \text{new violations})$ sinon
- Facteur phéromonal dépend de la stratégie :
  Stratégie vertex $\rightsquigarrow \tau_{factor}(\langle X_i, v_i\rangle, A) = \tau_{\langle X_i, v_i\rangle}$
  Stratégie clique $\rightsquigarrow \tau_{factor}(\langle X_i, v_i\rangle, A) = \sum_{\langle X_k, v_k\rangle \in A} \tau(\langle X_i, v_i\rangle, \langle X_k, v_k\rangle)$

Exemple 2 : Résultats expérimentaux

CSP solver competition in 2006

- 1195 binary instances defined in extension...
- ...selection of satisfiable instances
- ...selection of instances that pass through my parser

$\rightsquigarrow$ 230 instances from 6 benchmarks

Example 2 : ACO experimental setting

Hybridization with local search

At the end of each assignment construction, perform local search

- Move = change the value of one variable
- Accept the first non tabu and non deteriorating move
- Tabu list of infinite length

Parameter setting

- Choice of a setting that favors a quick convergence:
  $\alpha = 2$, $\beta = 8$, $\rho = 0.02$, 15 ants, $\tau_{min} = 0.01$, $\tau_{max} = \delta_{avg}/\rho$
  (where δ_{avg} = average qty of pheromone laid at each cycle)
- Exploitation of the resampling ratio to prevent from stagnation
  - When a solution is resampled, decrease pheromone trails on its components (multiplication by 0.1)
  - Restart if more than 1000 resamplings

Example 2 : considered solvers

- 23 solvers of the competition, all based on complete approaches
  $\rightsquigarrow$ CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO (Vertex and Clique)
  $\rightsquigarrow$ CPU time limit of 1800s on a 2.16GHZ Intel Core Duo
- Tabu (tabu list length=50, restart every $1,000,000$ moves)
  $\rightsquigarrow$ CPU time limit of 1800s on a 2.16GHZ Intel Core Duo

Solver	#solved
buggy_2_5_s	207
buggy_2_5	207
Abscon 109 AC	207
VALCSP 3.1	206
VALCSP 3.0	206
Abscon 109 ESAC	204
...	...

6 best complete solvers of the competition in 2006

Example 2 : considered solvers

- 23 solvers of the competition, all based on complete approaches
  $\rightsquigarrow$ CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO (Vertex and Clique)
  $\rightsquigarrow$ CPU time limit of 1800s on a 2.16GHZ Intel Core Duo
- Tabu (tabu list length=50, restart every $1,000,000$ moves)
  $\rightsquigarrow$ CPU time limit of 1800s on a 2.16GHZ Intel Core Duo

Solver	#solved
buggy_2_5_s	207
buggy_2_5	207
Abscon 109 AC	207
VALCSP 3.1	206
VALCSP 3.0	206
Abscon 109 ESAC	204
...	...
ACO(Vertex)	218
ACO(Clique)	214

6 best complete solvers of the competition in 2006

Example 2 : considered solvers

- 23 solvers of the competition, all based on complete approaches
  $\rightsquigarrow$ CPU time limit of 1800s on a 3GHZ Intel Xeon
- ACO (Vertex and Clique)
  $\rightsquigarrow$ CPU time limit of 1800s on a 2.16GHZ Intel Core Duo
- Tabu (tabu list length=50, restart every $1,000,000$ moves)
  $\rightsquigarrow$ CPU time limit of 1800s on a 2.16GHZ Intel Core Duo

Solver	#solved
buggy_2_5_s	207
buggy_2_5	207
Abscon 109 AC	207
VALCSP 3.1	206
VALCSP 3.0	206
Abscon 109 ESAC	204
...	...
ACO(Vertex)	218
ACO(Clique)	214
Tabu	226

6 best complete solvers of the competition in 2006

Example 2 : Experimental results

Easy benchmarks (all instances solved in less than 1800 s.)

- composed-25-10-20 : 10 instances, $|X| = 105$, $|D| = 10$
- marc : 5 instances, $|X| \in \{80, 84, 88, 92, 96\}$, $|D| = |X|$
- rand-2-k : 23 instances, $|X| = k$, $|D| = k$, $k \in \{23, 24, 25, 26, 27\}$

Benchmark	Best complete solver		ACO		Tabu
			Vertex	Clique	
comp-25-10-20	**0.06**	(VALCSP 3.1)	1.31	8.03	0.32
marc	**19.95**	(Abscon 109 AC)	26.96	27.05	28.21
rand-2-23	14.82	(VALCSP 3.0)	**1.42**	3.15	5.76
rand-2-24	9.94	(VALCSP 3.1)	**3.22**	7.43	8.25
rand-2-25	52.69	(buggy_2_5_s)	**3.20**	7.27	6.71
rand-2-26	180.86	(VALCSP 3.1)	**3.26**	87.71	21.13
rand-2-27	248.71	(VALCSP 3.1)	**23.45**	193.88	26.35

(average CPU time, in seconds, for each class of instances)

Example 2 : Experimental results

More difficult benchmarks

- rand-2-n-k-fcd : 60 instances, $|X| = n$, $|D| = k$
- geom : 92 instances, $|X|$=50, $|D| = 20$

	Best complete solver			ACO(Vertex)		ACO(Clique)		Tabu	
	nb	time	solver	nb	time	nb	time	nb	time
2-30-15	**20**	**0.2**	(VALCSP 3.0)	**20**	0.4	**20**	0.8	**20**	0.5
2-40-19	**20**	25.3	(VALCSP 3.1)	**20**	**7.0**	**20**	98.3	**20**	8.2
2-50-23	14	829.7	(buggy_2_5)	**20**	81.9	17	347.6	**20**	**31.0**
geom	**92**	**1.0**	(VALCSP 3.0)	91	11.0	91	50.2	**92**	3.6

nb = number of instances solved in less than 1800 s.
time = average CPU time for the solved instances

Example 2 : Experimental results

Really hard benchmark

- frb$n$-k [Xu et al. / IJCAI'05] : 40 instances, $|X| = n$, $|D| = k$

	Best solver			ACO(Vertex)		ACO(Clique)		Tabu	
	nb	time	solver	nb	time	nb	time	nb	time
30-15	**5**	**0.2**	(VALCSP 3.1)	**5**	0.4	**5**	1.3	**5**	0.5
35-17	**5**	2.1	(VALCSP 3.1)	**5**	3.0	**5**	5.0	**5**	**0.9**
40-19	**5**	9.4	(VALCSP 3.0)	**5**	**7.0**	**5**	103.5	**5**	9.1
45-21	**5**	123.4	(VALCSP 3.1)	**5**	467.7	**5**	354.1	**5**	**43.4**
50-23	3	327.2	(VALCSP 3.1)	3	430.4	3	680.5	**4**	**9.9**
53-24	1	74.0	(Abs. 109 AC)	3	105.7	3	530.8	**4**	**291.6**
56-25	0	-	-	2	535.8	2	170.2	**4**	**329.3**
59-26	1	413.9	(Abs. 109 AC)	1	63.6	0	-	**4**	**523.7**

nb = number of instances solved in less than 1800 s.
time = average CPU time for the solved instances