Cours de Master Recherche
Spécialité CODE :
Résolution de problèmes combinatoires

Christine Solnon

LIRIS, UMR 5205 CNRS / Université Lyon 1

2008

Approches incomplètes

Contexte

- Approches pour résoudre des pb d'optimisation $P = (E, f)$
  - E = Ensemble des combinaisons candidates
    $\rightsquigarrow$ Espace de recherche
  - $f : E \rightarrow \mathbb{R}$ = Fonction objectif à maximiser (ou minimiser)

  $\rightsquigarrow$ But = chercher $e^* \in E$ tel que $f(e^*)$ soit maximal
- Résolution de problèmes de satisfaction
  $\rightsquigarrow$ recherche de la combinaison « maximisant la satisfaction »

Caractéristiques des approches incomplètes

- Exploration opportuniste de l'espace de recherche E
  - **Intensifier** la recherche autour des zones prometteuses
  - **diversifier** la recherche pour explorer de nouvelles zones
- Pas de garantie d'optimalité... mais complexité polynomiale
  $\rightsquigarrow$ trouvent rapidement de bonnes combinaisons
- Approches anytime $\rightsquigarrow$ qualité améliorée au fil du temps

Approches incomplètes

Contexte

- Approches pour résoudre des pb d'optimisation $P = (E, f)$
  - E = Ensemble des combinaisons candidates
    $\rightsquigarrow$ Espace de recherche
  - $f : E \rightarrow \mathbb{R}$ = Fonction objectif à maximiser (ou minimiser)

  $\rightsquigarrow$ But = chercher $e^* \in E$ tel que $f(e^*)$ soit maximal
- Résolution de problèmes de satisfaction
  $\rightsquigarrow$ recherche de la combinaison « maximisant la satisfaction »

Caractéristiques des approches incomplètes

- Exploration opportuniste de l'espace de recherche E
  - **Intensifier** la recherche autour des zones prometteuses
  - **diversifier** la recherche pour explorer de nouvelles zones
- Pas de garantie d'optimalité... mais complexité polynomiale
  $\rightsquigarrow$ trouvent rapidement de bonnes combinaisons
- Approches anytime $\rightsquigarrow$ qualité améliorée au fil du temps

Deux familles d'approches incomplètes

Approches basées sur le voisinage

Nouvelles combinaisons construites à partir de comb. existantes

- modifications élémentaires $\rightsquigarrow$ recherche locale
- déplacement / vitesse $\rightsquigarrow$ essaims de particules
- croisements et mutations $\rightsquigarrow$ algorithmes génétiques

Approches constructives

Nouvelles combinaisons construites à l'aide de modèles

- modèles gloutons
- modèles gloutons aléatoires
- modèles évolutifs basés sur les distributions $\rightsquigarrow$ EDA
- modèles évolutifs basés sur la phéromone $\rightsquigarrow$ ACO

Deux familles d'approches incomplètes

Approches basées sur le voisinage

Nouvelles combinaisons construites à partir de comb. existantes

- modifications élémentaires $\rightsquigarrow$ recherche locale
- déplacement / vitesse $\rightsquigarrow$ essaims de particules
- croisements et mutations $\rightsquigarrow$ algorithmes génétiques

Approches constructives

Nouvelles combinaisons construites à l'aide de modèles

- modèles gloutons
- modèles gloutons aléatoires
- modèles évolutifs basés sur les distributions $\rightsquigarrow$ EDA
- modèles évolutifs basés sur la phéromone $\rightsquigarrow$ ACO

Approches incomplètes basées sur le voisinage

Principe général

- Génération de 1 ou plusieurs combinaisons initiales
  $\rightsquigarrow$ aléatoirement ou à l'aide d'un modèle
- **Tant que** qualité insuffisante **et** temps max non atteint :
  - Sélectionner 1 ou plusieurs combinaisons existantes
  - Créer 1 ou plusieurs comb. à partir de ces comb.

Trois instanciations de ce principe général

- Recherche locale :
  - Sélection de la dernière combinaison créée
  - Création d'une combinaison par mouvement
- Optimisation par essaims de particules (PSO)
  - Création de combinaisons par déplacement / vitesse
  - Mise-à-jour de la vitesse
- Algorithmes génétiques
  - Sélection des meilleures combinaisons de la population
  - Création de combinaisons par croisement + mutation

Approches incomplètes basées sur le voisinage

Principe général

- Génération de 1 ou plusieurs combinaisons initiales
  $\rightsquigarrow$ aléatoirement ou à l'aide d'un modèle
- **Tant que** qualité insuffisante **et** temps max non atteint :
  - Sélectionner 1 ou plusieurs combinaisons existantes
  - Créer 1 ou plusieurs comb. à partir de ces comb.

Trois instanciations de ce principe général

- Recherche locale :
  - Sélection de la dernière combinaison créée
  - Création d'une combinaison par mouvement
- Optimisation par essaims de particules (PSO)
  - Création de combinaisons par déplacement / vitesse
  - Mise-à-jour de la vitesse
- Algorithmes génétiques
  - Sélection des meilleures combinaisons de la population
  - Création de combinaisons par croisement + mutation

Graphe de voisinage

Définition d'un graphe de voisinage $G = (E, V)$

- Soit un espace de recherche E = ensemble de combinaisons
- Soit une approche incomplète basée sur le voisinage A
- Graphe de voisinage $G = (E, V)$ induit par A pour E :
  $V = \{(e_i, e_j) \ / \ A$ peut visiter e_j à partir de $e_i\}$
  $\rightsquigarrow$ e_j est une combinaison voisine de e_i
- Notation pour le voisinage d'une combinaison e_i
  $V(e_i) = \{e_j \ / \ (e_i, e_j) \in V\}$

Propriétés d'un graphe de voisinage $G = (E, V)$

- En général, G n'est pas orienté (mouvements symétriques)
- G doit être connexe
  $\rightsquigarrow$ possibilité d'atteindre n'importe quelle combinaison

Graphe de voisinage

Définition d'un graphe de voisinage $G = (E, V)$

- Soit un espace de recherche E = ensemble de combinaisons
- Soit une approche incomplète basée sur le voisinage A
- Graphe de voisinage $G = (E, V)$ induit par A pour E :
  $V = \{(e_i, e_j) \ / \ A$ peut visiter e_j à partir de $e_i\}$
  $\rightsquigarrow$ e_j est une combinaison voisine de e_i
- Notation pour le voisinage d'une combinaison e_i
  $V(e_i) = \{e_j \ / \ (e_i, e_j) \in V\}$

Propriétés d'un graphe de voisinage $G = (E, V)$

- En général, G n'est pas orienté (mouvements symétriques)
- G doit être connexe
  $\rightsquigarrow$ possibilité d'atteindre n'importe quelle combinaison

Ex. de voisinages : le voyageur de commerce

Rappel du problème

Soit un graphe $G = (S, A)$ et une fonction $d : A \rightarrow \mathbb{R}$

- Espace de recherche $E =$ ensemble des permutations de S
- Objectif : min. somme des distances des arêtes

Quels voisinages ?

Exemples de voisinages : la clique maximum

Rappel du problème

Soit un graphe $G = (S, A)$ et une fonction $d : A \rightarrow \mathbb{R}$

- Espace de recherche $E \subseteq \wp(S) =$ ens. des cliques de S
- Objectif : maximiser $f(e_i) = |e_i|$

Quels voisinages ?

Exemples de voisinages : SAT

Rappel du problème

Soient n variables booléennes et p clauses

- Espace de recherche $E = \{\text{vrai}, \text{faux}\}^n$
- Objectif : maximiser $f(e_i) =$ nombre de clauses satisfaites par e_i

Quels voisinages ?

Exemples de voisinages : CSP

Rappel du problème

Soit un CSP (X, D, C)

- Espace de recherche $E = D(X_1) \times \ldots \times D(X_n)$ si $X = \{X_1, \ldots, X_n\}$
- Objectif : maximiser $f(e_i) =$ nombre de contraintes de C satisfaites par e_i

Quels voisinages ?

Définition d'un paysage de recherche

Définition d'un paysage de recherche :

un problème d'optimisation $P = (E, f)$

+ un graphe de voisinage $G = (E, V)$

Représentation graphique d'un paysage de recherche

- Chaque configuration de l'espace de recherche E = 1 point
- La fonction objectif f donne l'altitude des points
- Le voisinage V positionne les points dans les autres dimensions

Définition d'un paysage de recherche

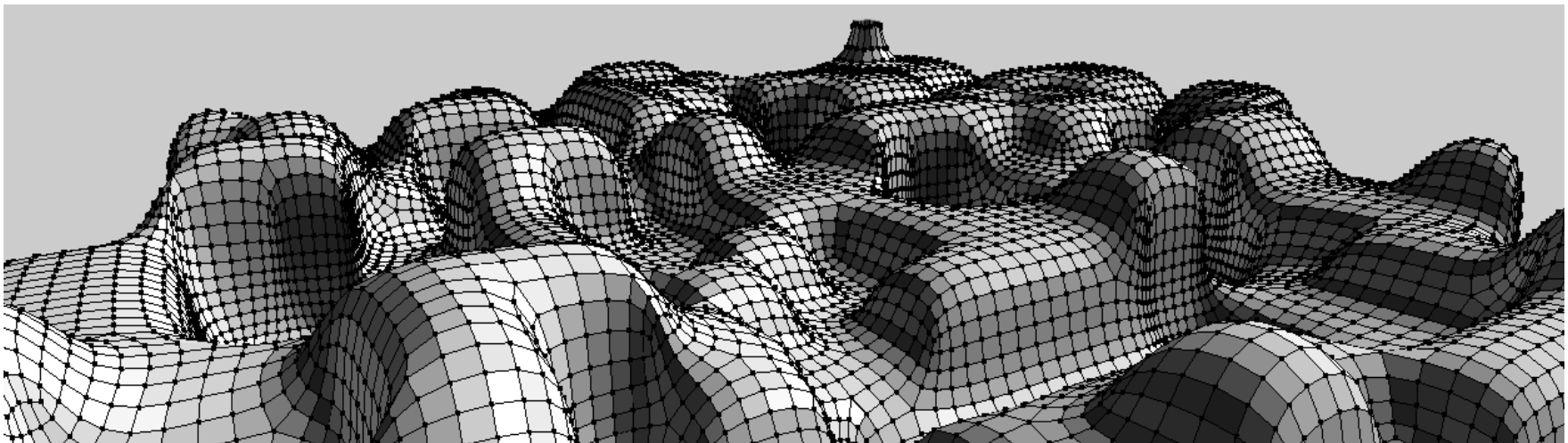
Définition d'un paysage de recherche :

un problème d'optimisation $P = (E, f)$

+ un graphe de voisinage $G = (E, V)$

Représentation graphique d'un paysage de recherche

- Chaque configuration de l'espace de recherche E = 1 point
- La fonction objectif f donne l'altitude des points
- Le voisinage V positionne les points dans les autres dimensions



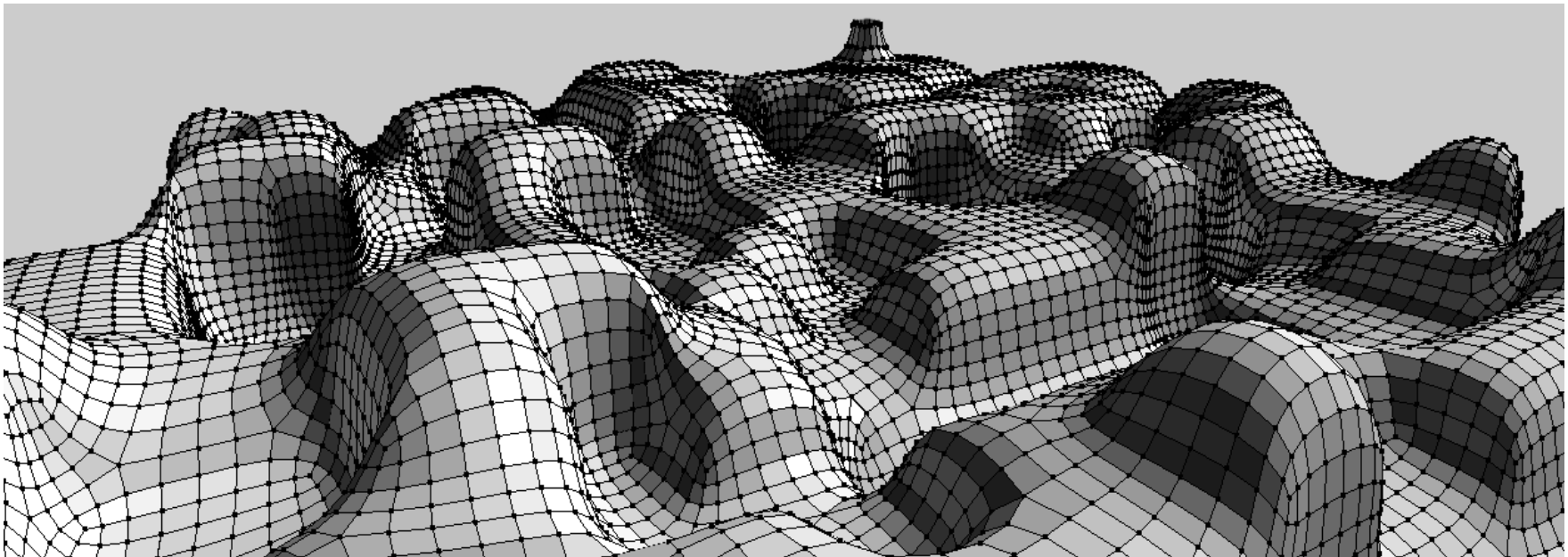
Optima locaux et plateaux (1)

Définitions

- Optimum local = point dont tous les voisins sont moins bons

$$e_i \in E \text{ tel que } \forall e_j \in V(e_i), f(e_j) < f(e_i)$$

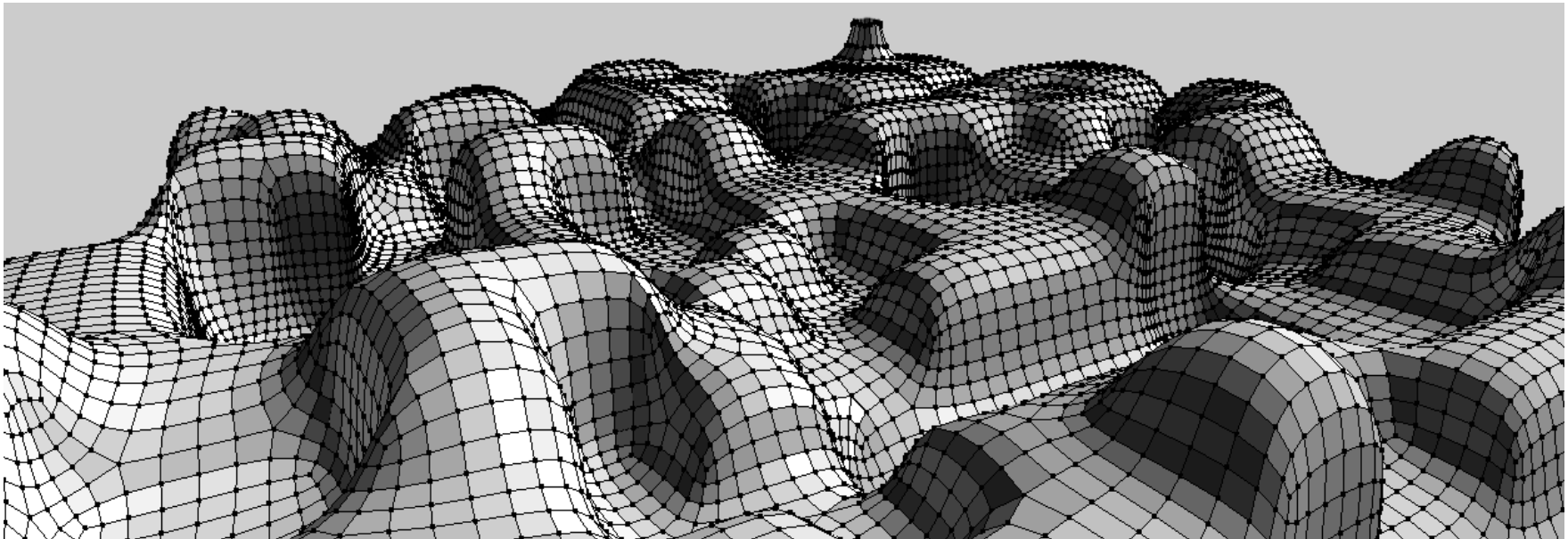
- Plateau = ensemble de points connexes dans le graphe $G = (E, V)$ ayant tous la même valeur pour f
- Bassin d'attraction d'un optimum local e
  $\rightsquigarrow$ points que l'on peut atteindre depuis e sans jamais monter



Optima locaux et plateaux (2)

Influence sur le choix d'une stratégie

- Paysage avec 1 seul optimum local et pas de plateau
  ⇝ Stratégie = intensifier
- Paysage rugueux = nombreux optima + distrib. uniforme
  ⇝ pas de corrélation entre qualité et distance à l'opt. global
  ⇝ Stratégie = diversifier
- Paysage de type "massif central"
  ⇝ Stratégie = intensifier et diversifier



Principe de base de la recherche locale

Algorithme Recherche Locale (E, V, f)

- $e \leftarrow$ une combinaison de E
- $e* \leftarrow e$
- **Tant que** $f(e*) <$ borne **et** temps max non atteint :
  - choisir $e' \in V(e)$
  - $e \leftarrow e'$
  - **Si** $f(e) > f(e*)$ **Alors** $e* \leftarrow e$
- **Retourner** $e*$

Principe de base de la recherche locale

Algorithme Recherche Locale (E, V, f)

- $e \leftarrow$ une combinaison de E
- $e* \leftarrow e$
- **Tant que** $f(e*) <$ borne **et** temps max non atteint :
  - choisir $e' \in V(e)$
  - $e \leftarrow e'$
  - **Si** $f(e) > f(e*)$ **Alors** $e* \leftarrow e$
- **Retourner** $e*$

Différentes stratégies possibles pour choisir un voisin

Recherche locale gloutonne (greedy local search)

(Aussi appelée « Hill climbing » et « montée de gradient »)

Stratégie pour choisir $e' \in V(e)$

$\rightsquigarrow$ choisir une combinaison $e' \in V(e)$ tq $f(e') > f(e)$ (ou $f(e') \geq f(e)$)

- Best improvement $\rightsquigarrow$ choisir le voisin qui maximise f
- First improvement $\rightsquigarrow$ choisir le premier voisin améliorant

- Intensification maximale $\rightsquigarrow$ converge sur un optimum local
  ...ou tourne en rond sur un plateau
- Peut être répété à partir de plusieurs combinaisons différentes
- Exemples d'algorithmes :
  - GSAT [Selman, Levesque, Mitchell 92] pour SAT
  - Min Conflict [Minton 92] pour les CSP

Recherche locale gloutonne (greedy local search)

(Aussi appelée « Hill climbing » et « montée de gradient »)

Stratégie pour choisir $e' \in V(e)$

$\rightsquigarrow$ choisir une combinaison $e' \in V(e)$ tq $f(e') > f(e)$ (ou $f(e') \geq f(e)$)

- Best improvement $\rightsquigarrow$ choisir le voisin qui maximise f
- First improvement $\rightsquigarrow$ choisir le premier voisin améliorant

- Intensification maximale $\rightsquigarrow$ converge sur un optimum local
  ...ou tourne en rond sur un plateau
- Peut être répété à partir de plusieurs combinaisons différentes
- Exemples d'algorithmes :
  - GSAT [Selman, Levesque, Mitchell 92] pour SAT
  - Min Conflict [Minton 92] pour les CSP

Recherche locale « random walk »

Stratégie pour choisir $e' \in V(e)$

$\rightsquigarrow$ Introduire la proba. p_{noise} de choisir un mouvement aléatoire

- Soit x un nombre tiré aléatoirement entre 0 et 1
- Si $x < p_{\text{noise}}$ Alors // diversification
  - Choisir aléatoirement e' dans $V(e)$

  Sinon // intensification
  - Choisir $e' \in V(e)$ qui maximise f

Paramètre p_{noise}

La valeur de p_{noise} détermine la diversification / intensification

- $p_{\text{noise}} = 1 \Rightarrow$ aléatoire pur
- $p_{\text{noise}} = 0 \Rightarrow$ glouton pur
- En général, $0.01 \leq p_{\text{noise}} \leq 0.1$

Recherche locale « random walk »

Stratégie pour choisir $e' \in V(e)$

$\rightsquigarrow$ Introduire la proba. p_{noise} de choisir un mouvement aléatoire

- Soit x un nombre tiré aléatoirement entre 0 et 1
- Si $x < p_{\text{noise}}$ Alors // diversification
  - Choisir aléatoirement e' dans $V(e)$

  Sinon // intensification
  - Choisir $e' \in V(e)$ qui maximise f

Paramètre p_{noise}

La valeur de p_{noise} détermine la diversification / intensification

- $p_{\text{noise}} = 1 \Rightarrow$ aléatoire pur
- $p_{\text{noise}} = 0 \Rightarrow$ glouton pur
- En général, $0.01 \leq p_{\text{noise}} \leq 0.1$

Recherche locale : « threshold accepting »

Stratégie pour choisir $e' \in V(e)$

- Introduire un seuil d'acceptation τ
- Choisir le premier voisin e' tel que $f(e') - f(e) > \tau$

Paramètre τ

La valeur de τ détermine la diversification / intensification

- $\tau \to -\infty \Rightarrow$ aléatoire pur
- $\tau \geq 0 \Rightarrow$ les mouvements dégradant la solution sont interdits

Le seuil d'acceptation τ peut augmenter au fil du temps
$\rightsquigarrow$ « gloutoniser » l'algorithme pour le faire converger

Recherche locale : « threshold accepting »

Stratégie pour choisir $e' \in V(e)$

- Introduire un seuil d'acceptation τ
- Choisir le premier voisin e' tel que $f(e') - f(e) > \tau$

Paramètre τ

La valeur de τ détermine la diversification / intensification

- $\tau \to -\infty \Rightarrow$ aléatoire pur
- $\tau \geq 0 \Rightarrow$ les mouvements dégradant la solution sont interdits

Le seuil d'acceptation τ peut augmenter au fil du temps
$\rightsquigarrow$ « gloutoniser » l'algorithme pour le faire converger

Recherche locale : « simulated annealing » (1)

Idée

- [Kirkpatrick 82] : inspiration = recuit simulé en métallurgie (équilibre énergétique lors de la cristallisation des métaux)
- Diversifier la recherche en autorisant des mouvements moins bons, en fonction d'une probabilité d'acceptation qui décroit avec le temps

Stratégie pour choisir $e' \in V(e)$

- Introduction d'un paramètre T qui décroit à chaque mouvement
- Choisir le premier voisin e' tel que
  - soit $f(e') \geq f(e)$ // intensification
  - soit $x < e^{-(f(e)-f(e'))/T}$ où x est un nombre choisi aléatoirement dans $[0; 1]$ // diversification

Recherche locale : « simulated annealing » (2)

Rôle de T $\rightsquigarrow$ diversification / intensification

- $T \to \infty \Rightarrow e^{-(f(e)-f(e'))/T} \to 1 \Rightarrow$ aléatoire pur
  $T \to 0 \Rightarrow e^{-(f(e)-f(e'))/T} \to 0 \Rightarrow$ glouton pur
- Au début, T grand $\rightsquigarrow$ Forte diversification
- Au fur et à mesure de la recherche, T diminue

  $\rightsquigarrow$ Intensification progressive autour des zones intéressantes
- Quand T proche de 0, système gelé sur un optimum local
- La vitesse avec laquelle la température diminue détermine le temps de convergence... et la qualité de la solution finale

Paramètres

- Valeur initiale de T ; Taux de diminution de T ; Seuil de gel
- Influence sur la qualité de la solution trouvée
- Paramétrage optimal dépend du pb à résoudre

Recherche locale : « simulated annealing » (2)

Rôle de T $\rightsquigarrow$ diversification / intensification

- $T \to \infty \Rightarrow e^{-(f(e)-f(e'))/T} \to 1 \Rightarrow$ aléatoire pur
  $T \to 0 \Rightarrow e^{-(f(e)-f(e'))/T} \to 0 \Rightarrow$ glouton pur
- Au début, T grand $\rightsquigarrow$ Forte diversification
- Au fur et à mesure de la recherche, T diminue

  $\rightsquigarrow$ Intensification progressive autour des zones intéressantes
- Quand T proche de 0, système gelé sur un optimum local
- La vitesse avec laquelle la température diminue détermine le temps de convergence... et la qualité de la solution finale

Paramètres

- Valeur initiale de T ; Taux de diminution de T ; Seuil de gel
- Influence sur la qualité de la solution trouvée
- Paramétrage optimal dépend du pb à résoudre

Recherche locale « Taboue » (1)

Idée

[Glover 86] : Toujours choisir le meilleur mouvement
... mais mémoriser les derniers mouvmts faits dans une liste taboue
... et interdire les mouvmts inverses afin de ne pas tourner en rond

Stratégie pour choisir $e' \in V(e)$

- Introduire une liste taboue l initialisée à vide
- A chaque itération, ajouter le dernier mouvement effectué dans l (mouvement = opération faite pour passer de e à e')
- Ne garder dans l que les k derniers mouvements effectués
- Choisir la combinaison $e' \in V(e)$ telle que
  - le mouvement $e \rightarrow e' \notin l$ // diversification
  - $f(e')$ soit maximal // intensification

Recherche locale « Taboue » (2)

- La liste taboue contient les k derniers mouvements effectués

  (... et non les k dernières solutions visitées)

  $\rightsquigarrow$ moins coûteux à vérifier et stocker

  Exemples :

  - Clique : mémoriser les sommets ajoutés/supprimés
  - SAT : mémoriser les variables « flippées »
- Possibilité d'ajouter un « critère d'aspiration » : accepter un mouvement tabou s'il permet d'obtenir une combinaison meilleure que $e*$
- k détermine la diversification / intensification
  - $k = 0 \Rightarrow$ Glouton pur
  - k petit $\rightsquigarrow$ risque de blocage dans un bassin d'attraction
  - Plus k augmente, et plus la recherche est diversifiée
    ... mais on risque de s'interdire de monter sur le pic optimal
- Là encore, le paramétrage est un point délicat...

Recherche locale « réactive » (1)

- La longueur de la liste Taboue est un point déterminant
  - Trop courte ⇒ Intensification trop forte
    ⇝ blocage de la recherche autour d'un optimum local
  - Trop longue ⇒ Diversification trop forte
    ⇝ la recherche risque de passer à coté des meilleures combinaisons
- La longueur optimale de la liste varie
  - d'un problème à l'autre
  - d'une instance à l'autre d'un même problème
  - au cours de la résolution d'une même instance
- [Battiti, Protasi 2001] : adapter cette longueur dynamiquement
  - Besoin de diversification ⇒ augmenter la longueur
  - Besoin d'intensification ⇒ diminuer la longueur

Recherche locale « réactive » (1)

- La longueur de la liste Taboue est un point déterminant
  - Trop courte ⇒ Intensification trop forte
    ⇝ blocage de la recherche autour d'un optimum local
  - Trop longue ⇒ Diversification trop forte
    ⇝ la recherche risque de passer à coté des meilleures combinaisons
- La longueur optimale de la liste varie
  - d'un problème à l'autre
  - d'une instance à l'autre d'un même problème
  - au cours de la résolution d'une même instance
- [Battiti, Protasi 2001] : adapter cette longueur dynamiquement
  - Besoin de diversification ⇒ augmenter la longueur
  - Besoin d'intensification ⇒ diminuer la longueur

Recherche locale « réactive » (1)

- La longueur de la liste Taboue est un point déterminant
  - Trop courte ⇒ Intensification trop forte
    ⇝ blocage de la recherche autour d'un optimum local
  - Trop longue ⇒ Diversification trop forte
    ⇝ la recherche risque de passer à coté des meilleures combinaisons
- La longueur optimale de la liste varie
  - d'un problème à l'autre
  - d'une instance à l'autre d'un même problème
  - au cours de la résolution d'une même instance
- [Battiti, Protasi 2001] : adapter cette longueur dynamiquement
  - Besoin de diversification ⇒ augmenter la longueur
  - Besoin d'intensification ⇒ diminuer la longueur

Recherche locale « réactive » (2)

Principe

- Utiliser une table de Hachage pour mémoriser (en temps constant) les clés des combinaisons visitées
- Collision dans la table ⇒ besoin de diversification
  ⇝ allonger la liste taboue
- Longue période sans collision ⇒ besoin d'intensification
  ⇝ raccourcir la liste taboue

Paramètres de l'algorithme :

- Longueurs initiale, minimale et maximale de la liste
- Taille de l'allongement de la liste
- Taille de la période sans collision
- Taille du raccourcissement de la liste

⇝ Plus de paramètres que la recherche taboue non réactive
... mais paramétrage plus robuste en pratique

Recherche locale « réactive » (2)

Principe

- Utiliser une table de Hachage pour mémoriser (en temps constant) les clés des combinaisons visitées
- Collision dans la table $\Rightarrow$ besoin de diversification
  $\rightsquigarrow$ allonger la liste taboue
- Longue période sans collision $\Rightarrow$ besoin d'intensification
  $\rightsquigarrow$ raccourcir la liste taboue

Paramètres de l'algorithme :

- Longueurs initiale, minimale et maximale de la liste
- Taille de l'allongement de la liste
- Taille de la période sans collision
- Taille du raccourcissement de la liste

$\rightsquigarrow$ Plus de paramètres que la recherche taboue non réactive
... mais paramétrage plus robuste en pratique

Recherche à voisinage variable / Variable Neighborhood Search (VNS)

Motivation

- Plusieurs voisinages possibles pour un même problème
- Un opt. local pour un voisinage n'est pas opt. dans un autre vois.

$\rightsquigarrow$ changer de voisinage quand on arrive sur un opt. local

Stratégie pour choisir e' dans un voisinage de e

- Soient V^1, V^2, V^3, ... une suite de voisinages
  $V^t(e)$ = voisins de e pour le t^{ieme} voisinage
  En général, $|V^t(e)| > |V^{t-1}(e)|$
- $t \leftarrow 1$
- **Tant que** $max\{f(e'')/e'' \in V^t(e)\} \leq f(e)\} : t \leftarrow t+1$
- $e' \leftarrow$ meilleure combinaison de $V^t(e)$

Recherche à voisinage variable / Variable Neighborhood Search (VNS)

Motivation

- Plusieurs voisinages possibles pour un même problème
- Un opt. local pour un voisinage n'est pas opt. dans un autre vois.

$\rightsquigarrow$ changer de voisinage quand on arrive sur un opt. local

Stratégie pour choisir e' dans un voisinage de e

- Soient V^1, V^2, V^3, ... une suite de voisinages
  $V^t(e) =$ voisins de e pour le t^{ieme} voisinage
  En général, $|V^t(e)| > |V^{t-1}(e)|$
- $t \leftarrow 1$
- **Tant que** $max\{f(e'')/e'' \in V^t(e)\} \leq f(e)\} : t \leftarrow t + 1$
- $e' \leftarrow$ meilleure combinaison de $V^t(e)$

Recherche à très grand voisinage / Very Large Neighborhood Search (VLNS)

Idée

- Définir un voisinage très large
- Utiliser une approche complète pour explorer ce voisinage
  $\rightsquigarrow$ trouver le meilleur voisin
- Approche gloutonne $\rightsquigarrow$ intensification
- Très grand voisinage $\rightsquigarrow$ diversification

Recherche locale / plusieurs points de départ

Un processus de recherche locale peut être itéré plusieurs fois, à partir de points différents :

- Multi-start local search : Les points de départs sont indépendants
- Iterated local search : Le point de départ d'une recherche locale est une perturbation de la meilleure solution de la dernière recherche locale effectuée
- Go with the winner :
  - Plusieurs recherches locales en parallèle (particules)
  - Les moins bonnes sont ≪ redistribuées ≫ autour des meilleures
- Genetic local search (scatter search) :
  - Une population de solutions évolue par sélections/croisements/mutations
  - Chaque nouvelle solution générée est améliorée par recherche locale

Optimisation par essaims de particules

Basé sur la métaphore du déplacement en essaim / banc
$\rightsquigarrow$ déplacer un essaim de combinaisons dans E

- **Essaim initial** : Générer un ensemble P de n combinaisons, et une vitesse initiale v_i pour chaque combinaison $e_i \in P$
- **Tant que** qualité insuffisante **et** temps max non atteint :
  - Pour chaque combinaison $e_i \in P$ faire :
    - modifier la vitesse de e_i

$$v_i \leftarrow \omega v_i + c_1 f_1 (best_i - e_i) + c_2 f_2 (best_{V(e_i)} - e_i)$$

où ω = inertie, c_1 et c_2 = accélération,
et f_1 et f_2 = nombres aléatoires $\in [0; 1]$

    - déplacer e_i en fonction de v_i : $e_i \leftarrow e_i + v_i$

Pour en savoir plus

http://en.wikipedia.org/wiki/Particle_swarm_optimization

Algorithmes génétiques

Basés sur la métaphore de l'évolution des espèces
$\rightsquigarrow$ Faire évoluer une population de combinaisons

- **Population initiale** :
  Générer un ensemble P de n combinaisons
- **Tant que** qualité insuffisante **et** temps max non atteint :
  - **Sélection** d'un ensembre $P' \subseteq P$ de combinaisons
    $\rightsquigarrow$ Favoriser la sélection des meilleures combinaisons
    ... tout en préservant la diversité
  - **Croisement** des comb. de P'
    $\rightsquigarrow$ Création de nouvelles combinaisons P''
  - **Mutation** de quelques composants des comb. de P''
  - **Mise à jour** de P à l'aide de P''

Pour en savoir plus

http://en.wikipedia.org/wiki/Genetic_algorithm

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

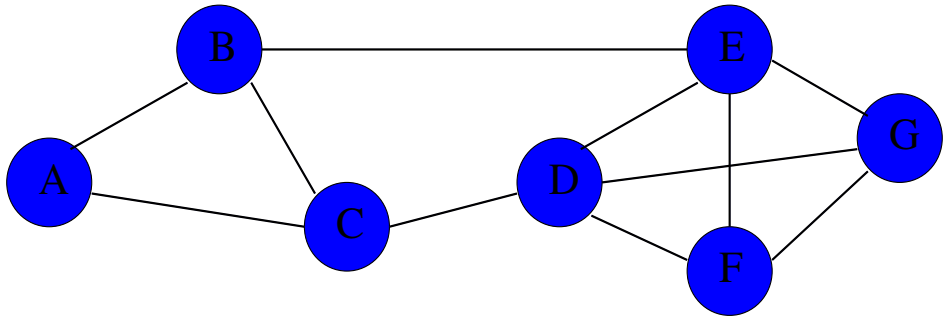
- Choisir aléatoirement un sommet $s_i \in S$
- $\pi =< s_i >$
- $Cand \leftarrow S - \{s_i\}$
- tant que $Cand \neq \emptyset$ faire
  - Choisir $s_j \in$ candidats selon une heuristique gloutonne
  - ajouter s_j à la fin de π
  - $Cand \leftarrow Cand - \{s_j\}$

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
  - $C \leftarrow C \cup \{s_j\}$
  - $Cand \leftarrow Cand \cap \{s_i / (s_i, s_j) \in A\}$

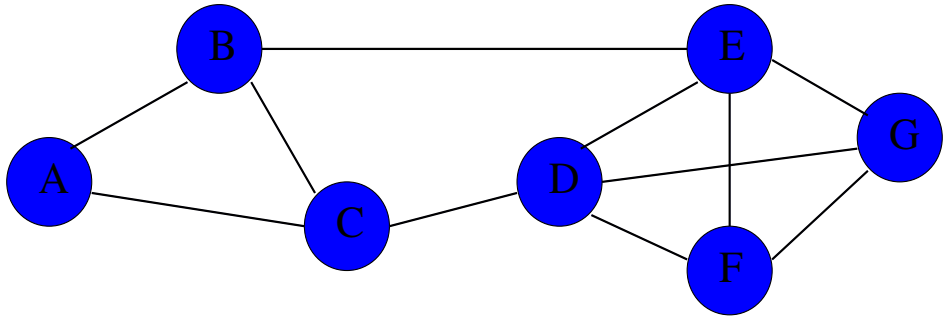
Quelle heuristique gloutonne ???



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

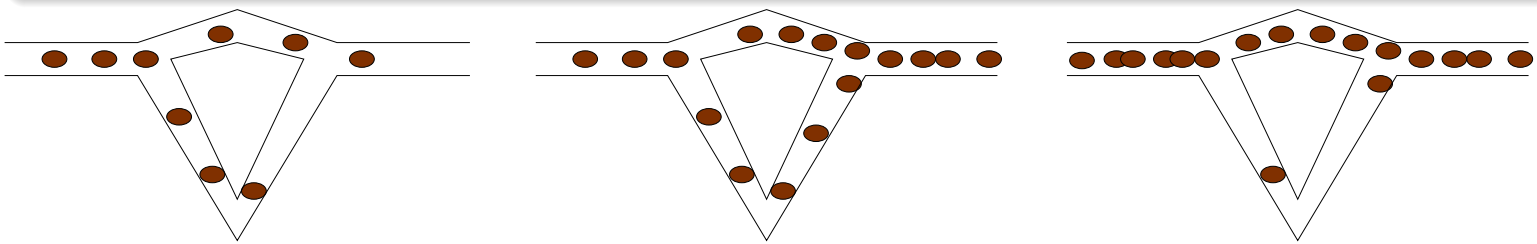
  - Algorithmes par estimation de distribution
    $\rightsquigarrow$ statistiques sur les expériences passées
  - Algorithmes à base de colonies de fourmis
    $\rightsquigarrow$ expérience passée “compilée” sous forme de phéromone

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

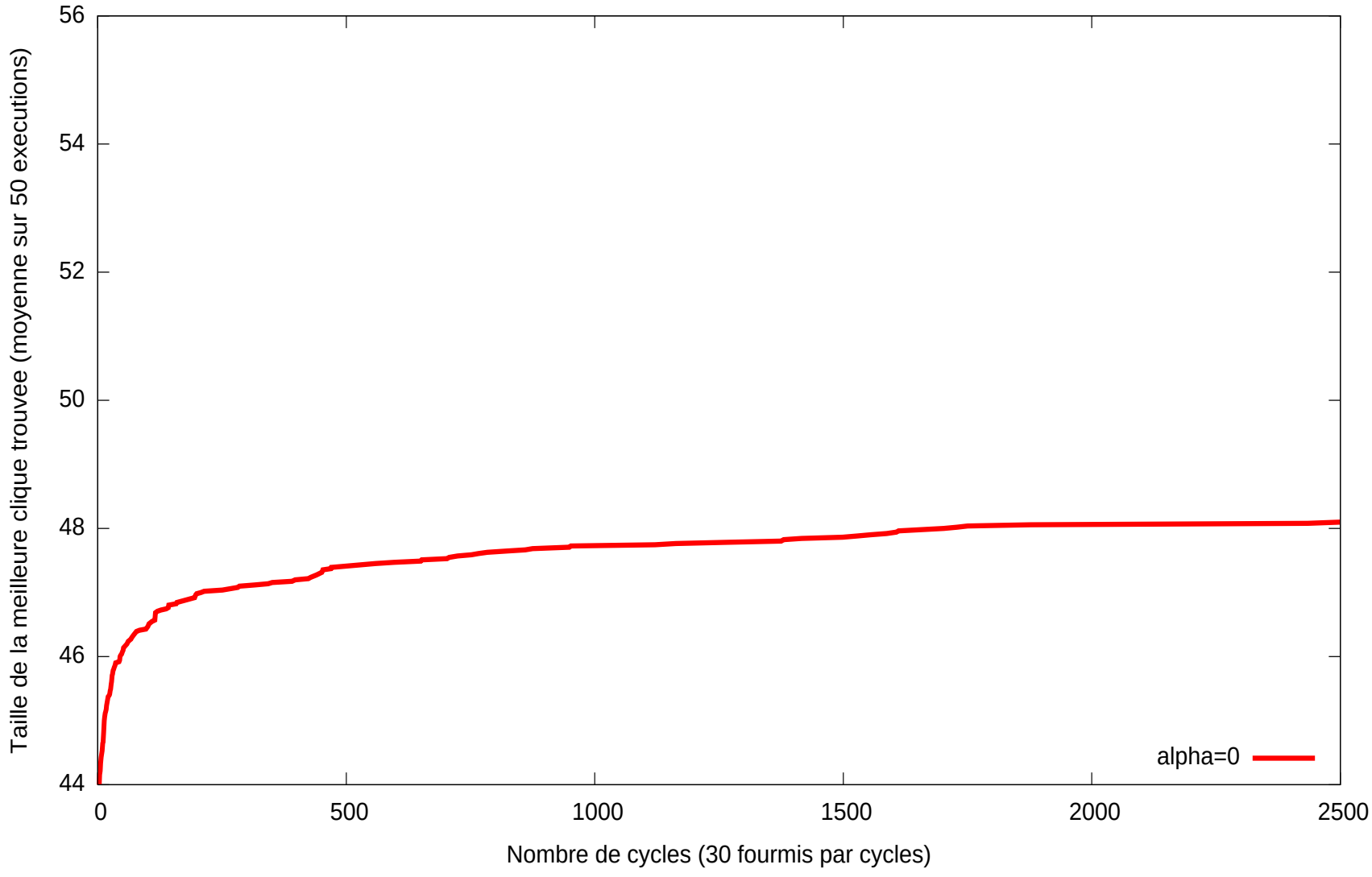
$\rightsquigarrow$ l'intensification augmente quand α augmente

ρ : taux d'évaporation

$\rightsquigarrow$ l'intensification augmente quand ρ augmente

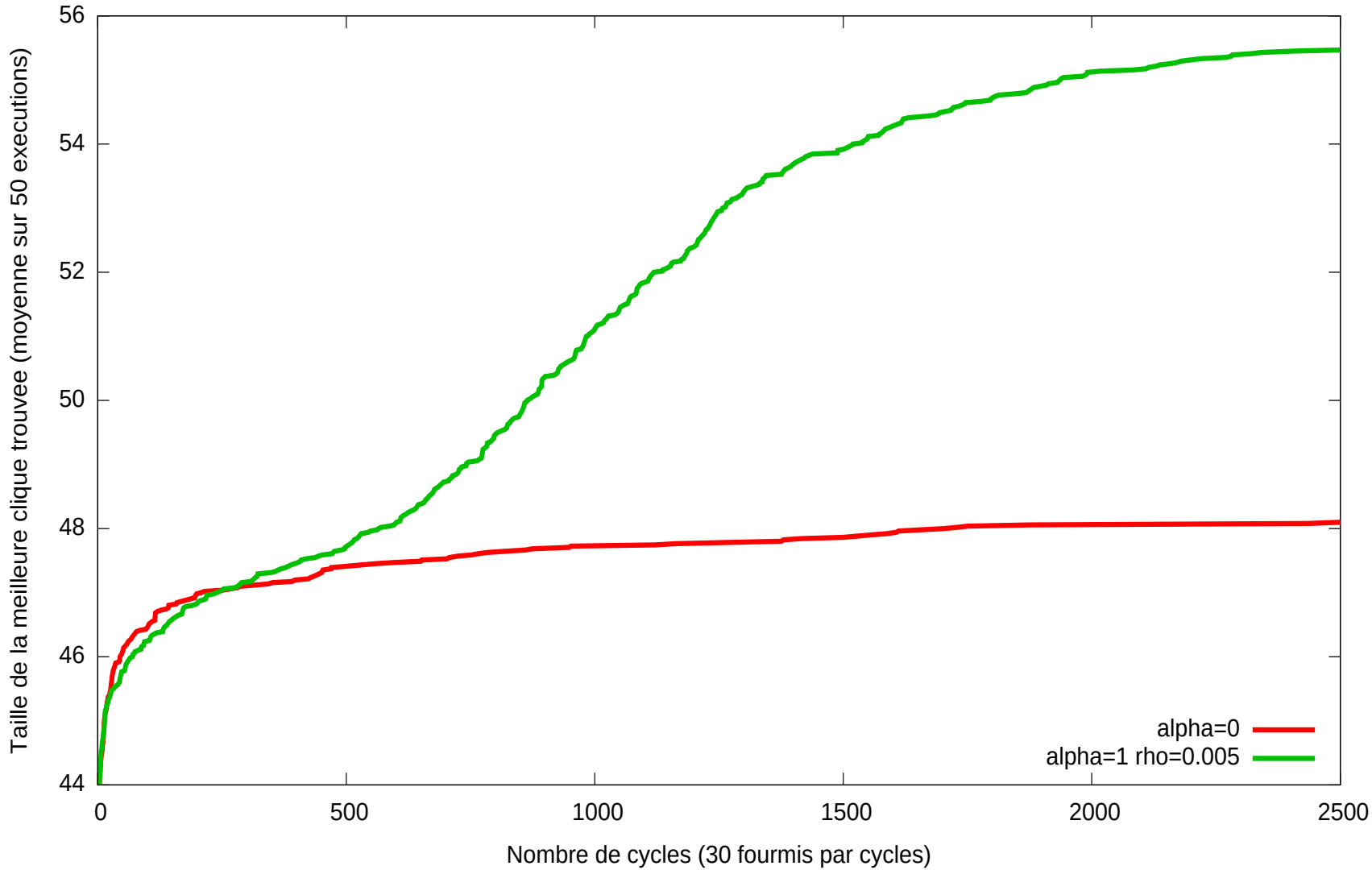
Le meilleur paramétrage dépend du temps disponible !

Illustration sur un problème de recherche de clique



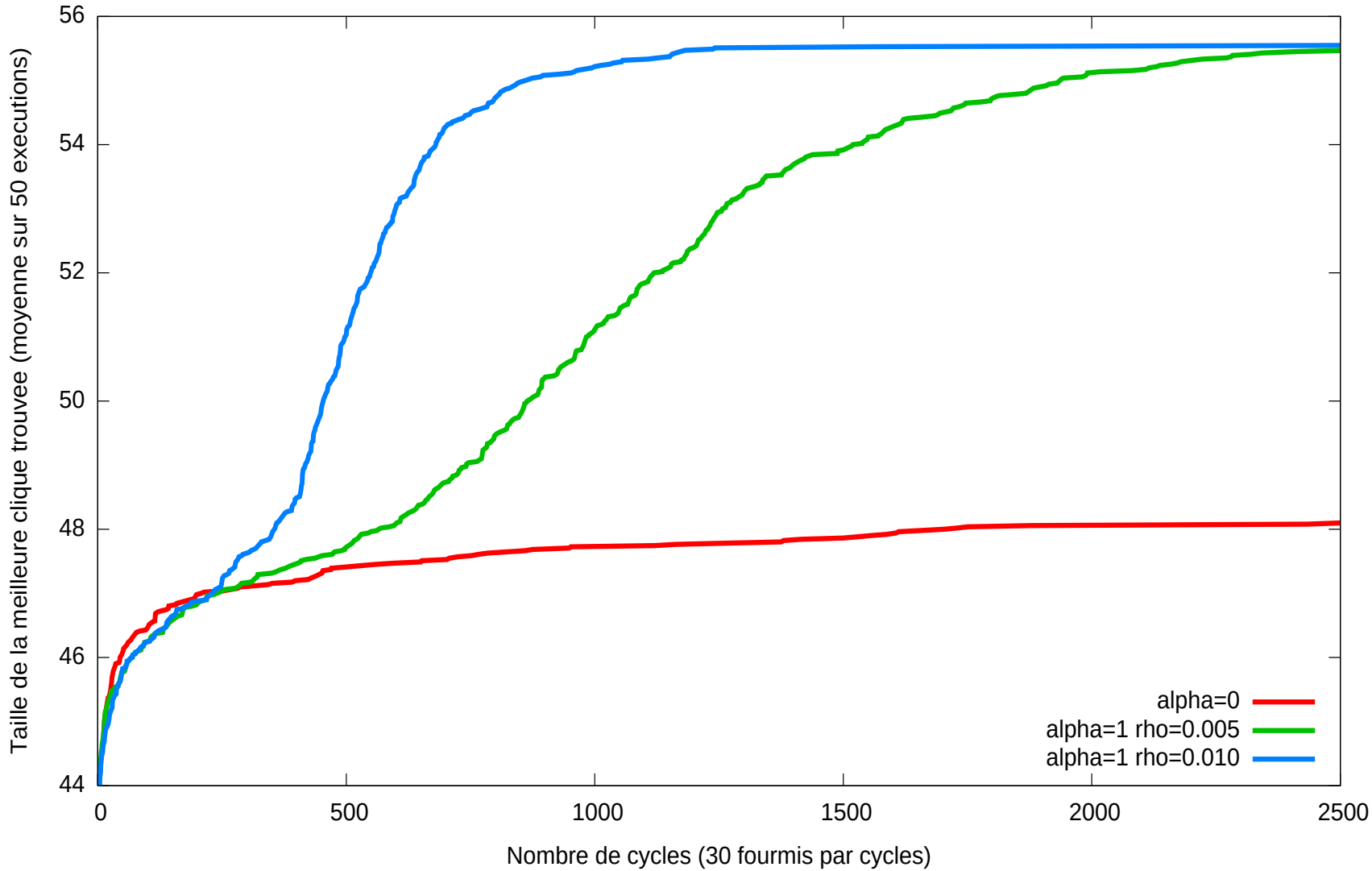
Sans phéromone : $\alpha = 0$

Illustration sur un problème de recherche de clique



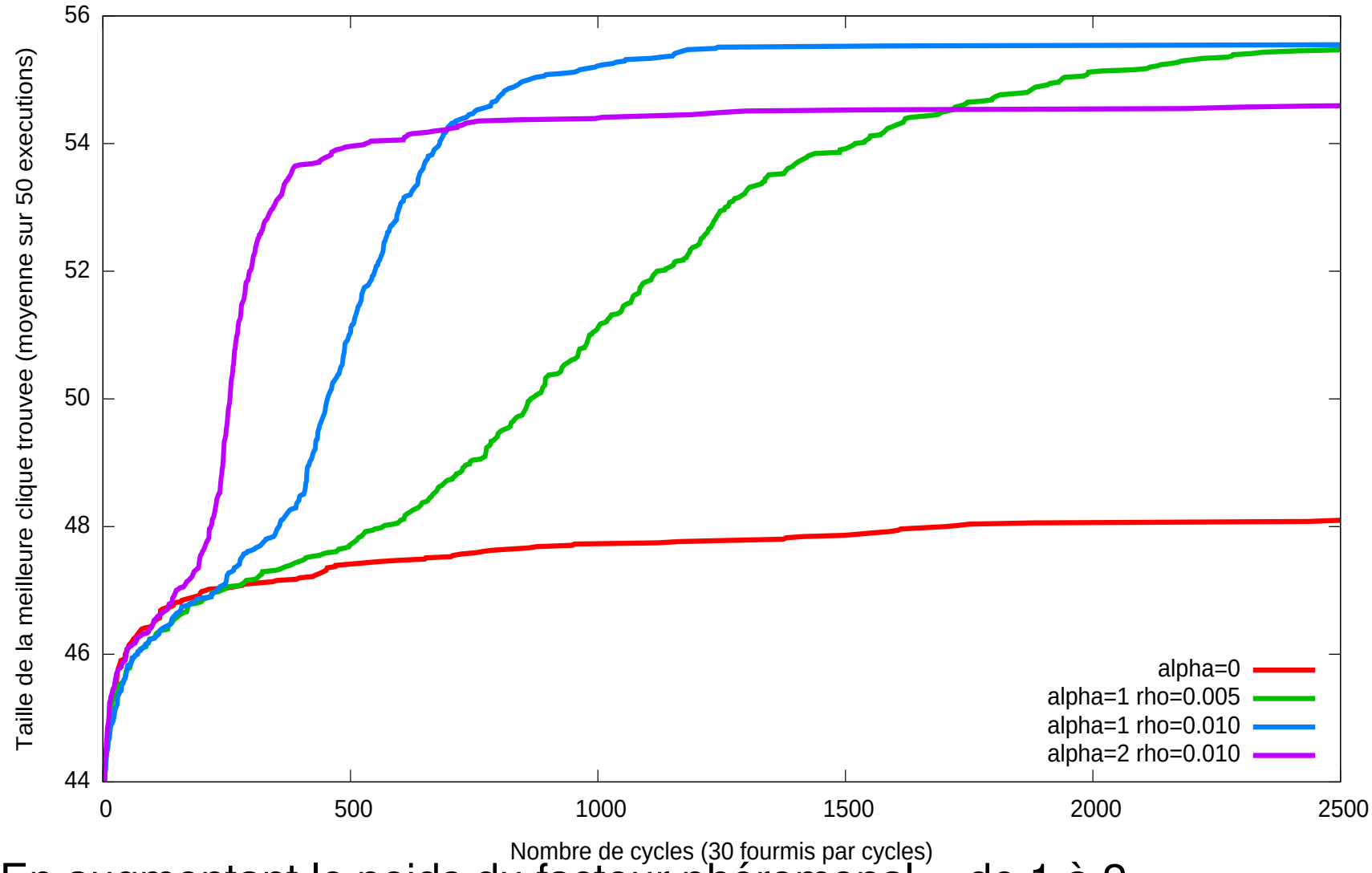
Avec une petite influence de la phéromone : $\alpha = 1$, $\rho = 0.005$

Illustration sur un problème de recherche de clique



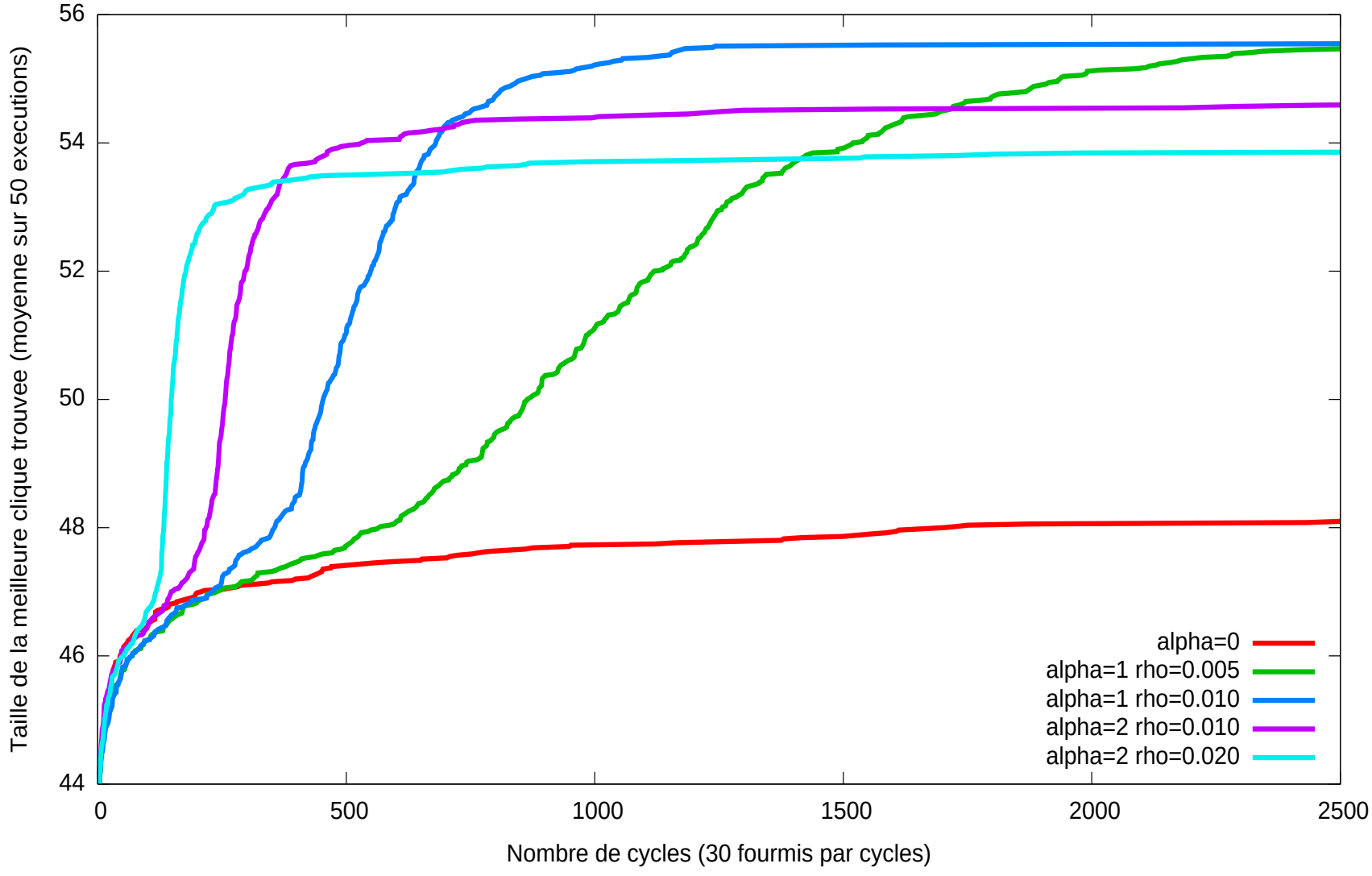
En augmentant l'évaporation ρ de 0.5 % à 1%

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

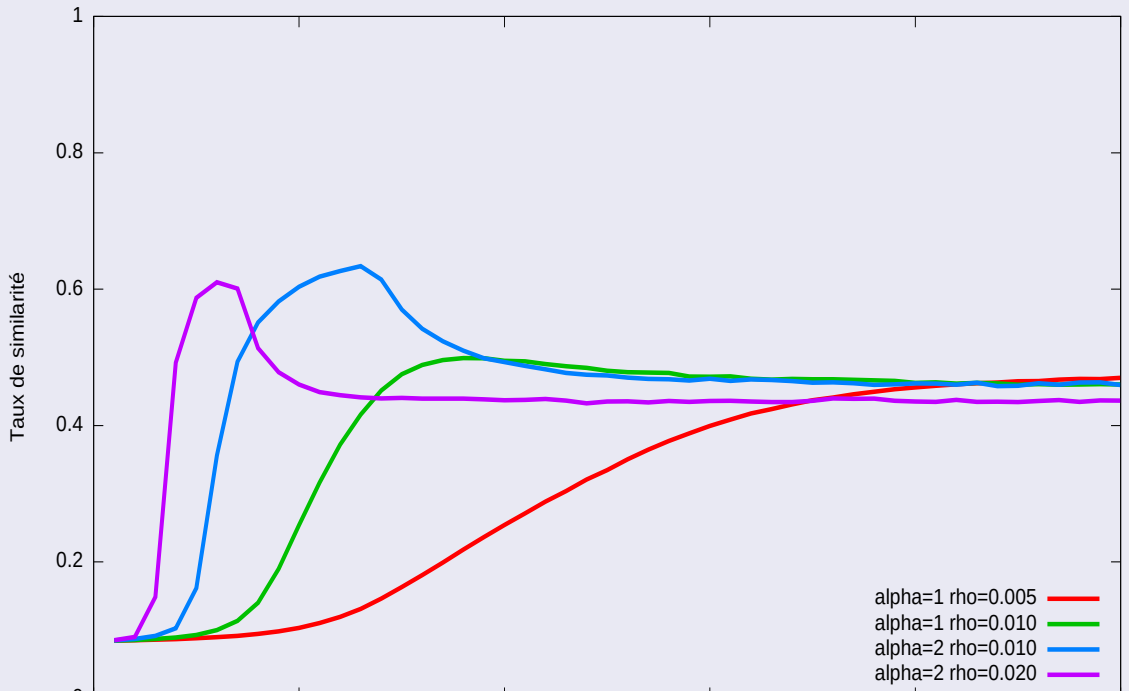
Calcul en temps (quasi) constant de ces deux indicateurs !!!

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



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

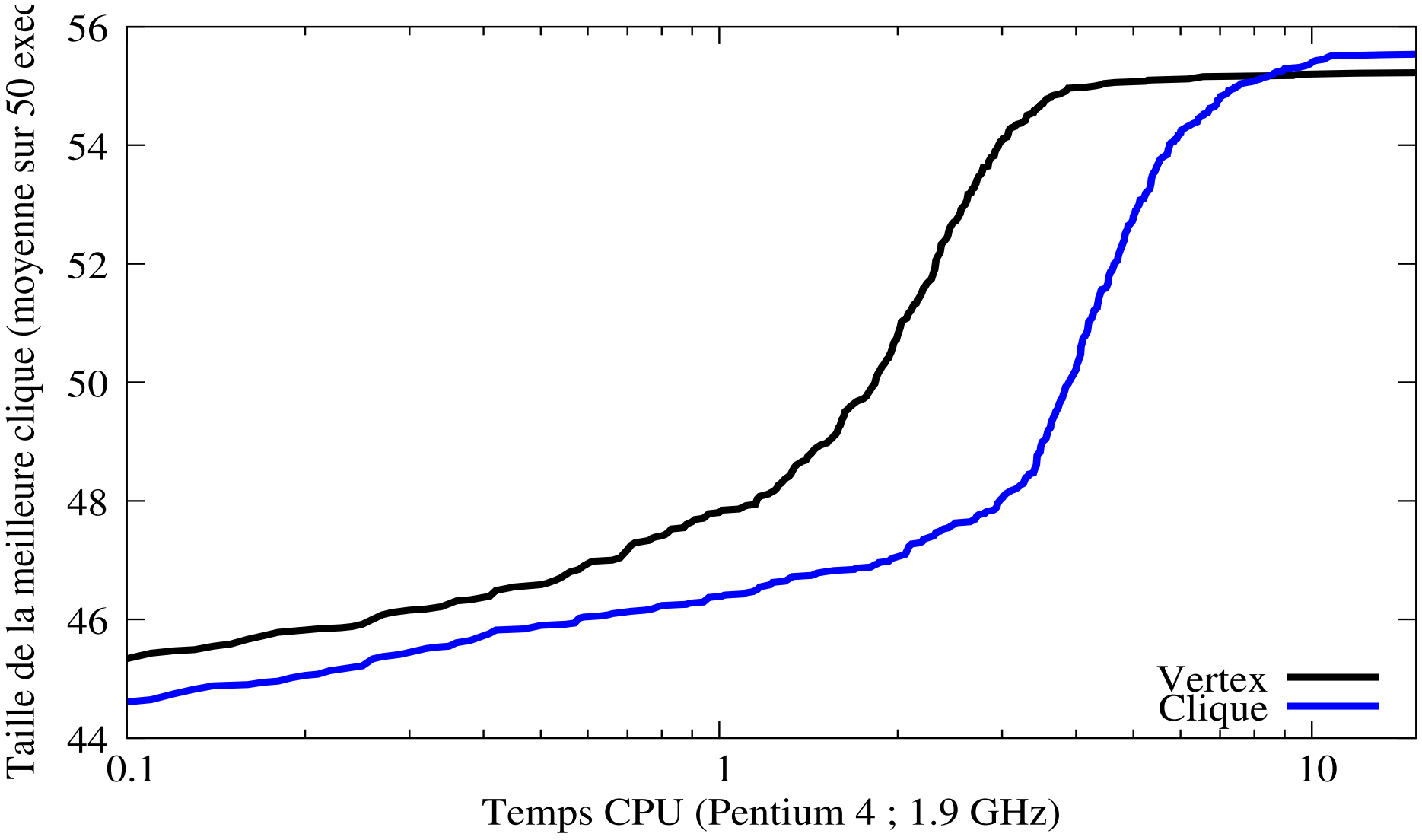
Exemple 1 : Clique maximum

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

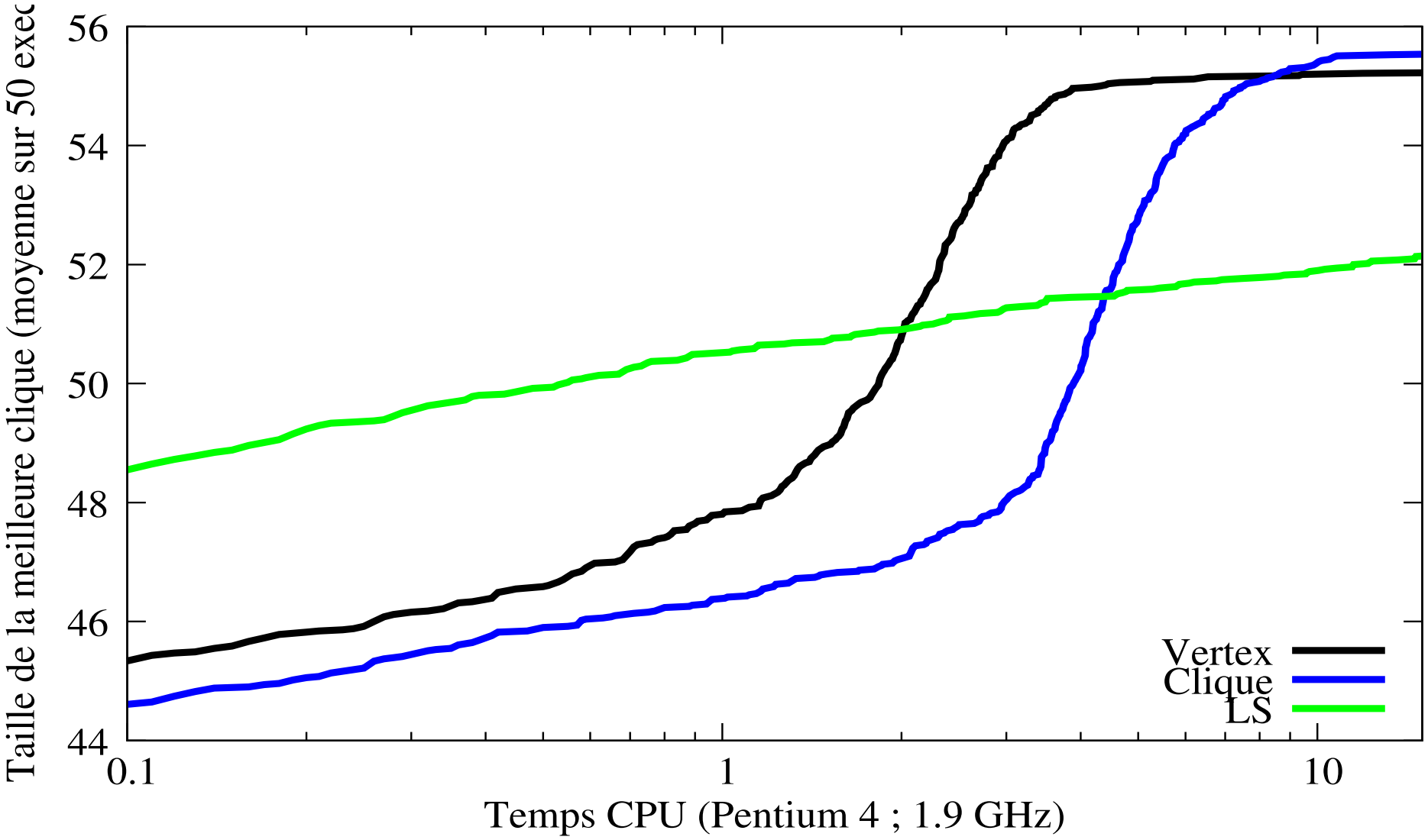
- Choix d'un algorithme de recherche locale

  $\rightsquigarrow$ Trouver un compromis entre temps et qualité

  $\rightsquigarrow$ En général : Recherche locale gloutonne

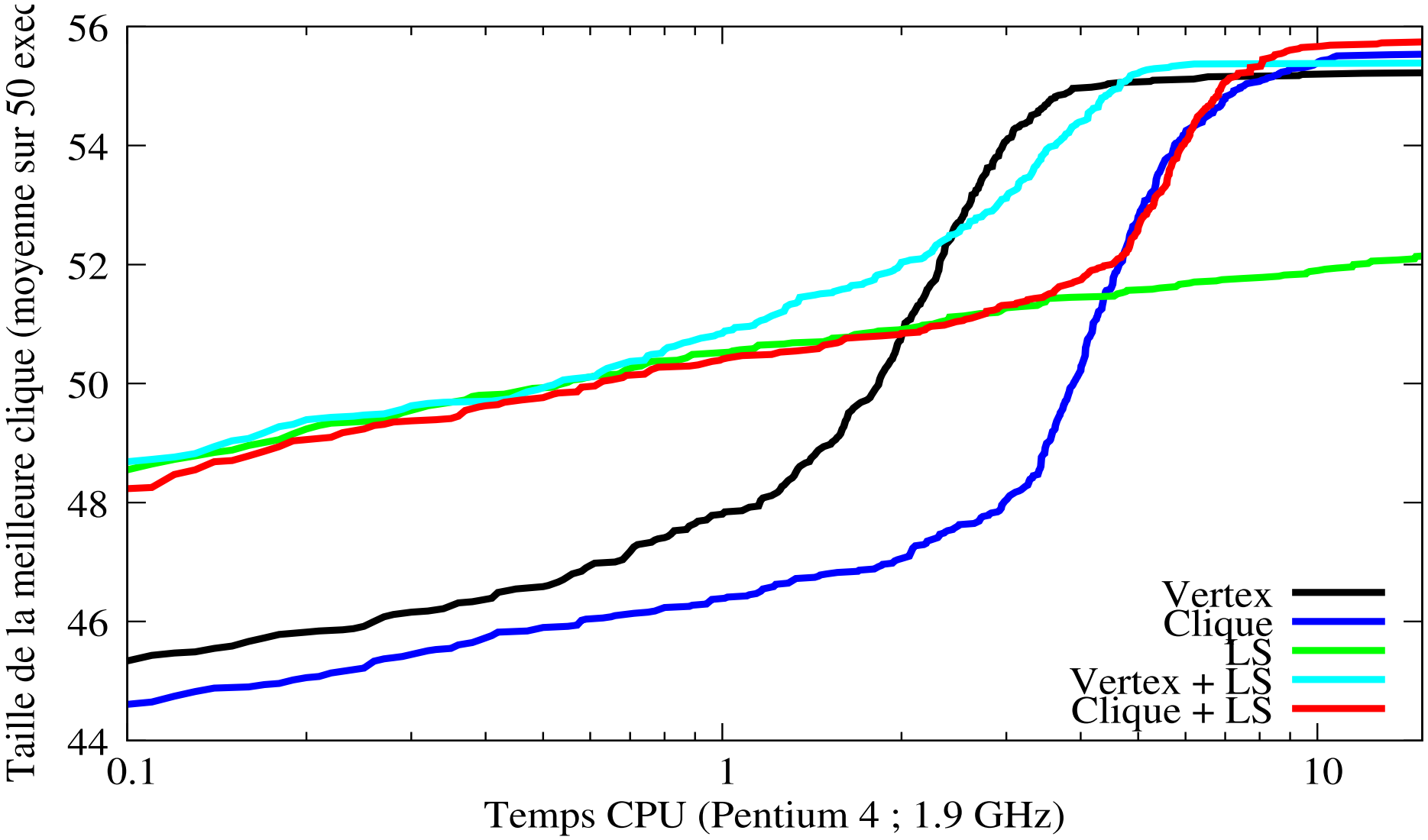
- ACO + recherche locale :
  - ACO $\rightsquigarrow$ calculer de nouveaux points de départ à partir des optima locaux trouvés par la recherche locale
  - Recherche locale $\rightsquigarrow$ amélioration des solutions calculées par ACO

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