Méta-heuristiques pour l'optimisation difficile

Cédric Buche

CERV

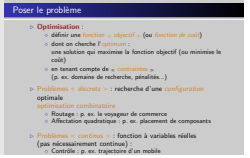
6 novembre 2012

Page 1 :

Après une introduction à la problématique de l'optimisation dite difficile, ce cours présente différentes familles d'heuristiques pour la résolution de problèmes continus et discontinus : recuit simulé, recherche avec tabou, colonies de fourmis, essaims particulaires...

Poser le problème

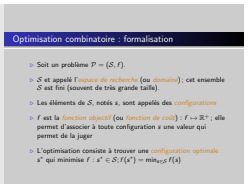
- **Optimisation** :
  - définir une fonction « *objectif* » (ou *fonction de coût*)
  - dont on cherche l'*optimum* :
    une solution qui maximise la fonction objectif (ou minimise le coût)
  - en tenant compte de « contraintes »
    (p. ex. domaine de recherche, pénalités...)
- Problèmes « *discrets* » : recherche d'une *configuration* optimale
  optimisation combinatoire
  - Routage : p. ex. le voyageur de commerce
  - Affectation quadratique : p. ex. placement de composants
- Problèmes « *continus* » : fonction à variables réelles (pas nécessairement continue) :



Page 2 :

Optimisation combinatoire : formalisation

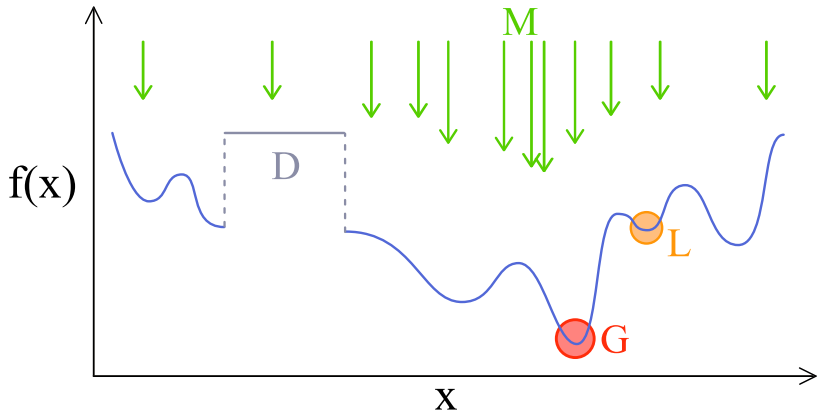
- Soit un problème $\mathcal{P} = (\mathcal{S}, f)$.
- $\mathcal{S}$ et appelé l'*espace de recherche* (ou *domaine*) ; cet ensemble $\mathcal{S}$ est fini (souvent de très grande taille).
- Les éléments de $\mathcal{S}$, notés s, sont appelés des *configurations*
- f est la *fonction objectif* (ou *fonction de coût*) : $f \mapsto \mathbb{R}^+$; elle permet d'associer à toute configuration s une valeur qui permet de la juger
- L'optimisation consiste à trouver une configuration optimale s^* qui minimise f : $s^* \in \mathcal{S}; f(s^*) = \min_{s \in \mathcal{S}} f(s)$

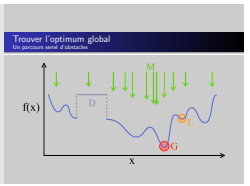


Page 3 :

Trouver l'optimum global

Un parcours semé d'obstacles





Page 4 :

Source de l'illustration : wikipedia.

Les difficultés

Taille des problèmes (complexité de la recherche)

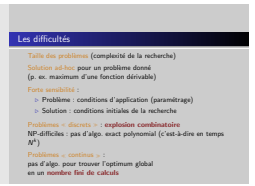
Solution ad-hoc pour un problème donné
(p. ex. maximum d'une fonction dérivable)

Forte sensibilité :

- Problème : conditions d'application (paramétrage)
- Solution : conditions initiales de la recherche

Problèmes « discrets » : **explosion combinatoire**
NP-difficiles : pas d'algo. exact polynomial (c'est-à-dire en temps N^k)

Problèmes « continus » :
pas d'algo. pour trouver l'optimum global
en un **nombre fini de calculs**



Page 5 :

On qualifie un problème de classe P s'il est de complexité Polynomial, c'est-à-dire en $O(n^k)$. La classe de complexité NP désigne les problèmes Non-déterministes Polynomiaux. Elle réunit les problèmes de décision qui peuvent être décidés par un calcul non déterministe en temps polynomial.
On dit qu'un problème est NP-complet ou NP-difficile s'il est au moins aussi difficile que tous les problèmes de la classe NP. Voir sur wikipedia la liste des problèmes NP-complets.
Un calcul est dit déterministe si le résultat suivant du calcul obtenu par un algorithme est déterminé par le résultat courant. Dans le cas contraire, il est dit non déterministe.

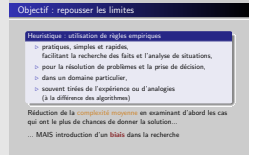
Objectif : repousser les limites

Heuristique : utilisation de règles empiriques

- pratiques, simples et rapides,
  facilitant la recherche des faits et l'analyse de situations,
- pour la résolution de problèmes et la prise de décision,
- dans un domaine particulier,
- souvent tirées de l'expérience ou d'analogies
  (à la différence des algorithmes)

Réduction de la complexité moyenne en examinant d'abord les cas qui ont le plus de chances de donner la solution...

... MAIS introduction d'un **biais** dans la recherche



Page 6 :

Repousser encore les limites

Métaheuristique

Méthode générale d'application d'heuristiques

Pas spécifique à un type donné de problème

Origines des métaheuristiques : **analogies**

- ▷ physique : recuit simulé
- ▷ biologie : algo. évolutionnaires, réseaux de neurones, systèmes immunitaires artificiels
- ▷ cognition humaine : recherche avec tabous, apprentissage artificiel
- ▷ éthologie : colonies de fourmis, essaims

Page 7 :

Des principes communs

dans les grandes lignes...

Solution initiale : $s_0 \mapsto f(s_0)$

Itération i : modification de la solution courante s_i :
choix d'un type de mouvement
application d'un *mouvement* m

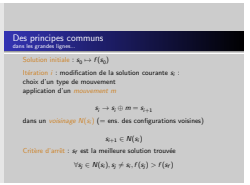
$$s_i \rightarrow s_i \oplus m = s_{i+1}$$

dans un *voisinage* $N(s_i)$ (= ens. des configurations voisines)

$$s_{i+1} \in N(s_i)$$

Critère d'arrêt : s_f est la meilleure solution trouvée

$$\forall s_j \in N(s_i), s_j \neq s_i, f(s_j) > f(s_f)$$

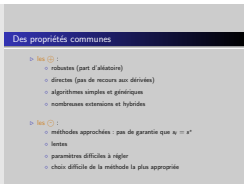


Page 8 :

Toute méta-heuristique repose (en partie) sur une recherche locale de solutions.
On appelle *voisinage* d'une configuration s un sous-ensemble de $\mathcal{S}$, noté $N(s)$ dont les éléments sont toutes les configurations s' que l'on peut obtenir en appliquant à s un *mouvement* m appartenant à un ensemble de mouvements $\mathcal{M}$. L'application d'un mouvement m à s est noté $s \oplus m$. Le voisinage se définit alors comme : $N(s) = \{s' \in \mathcal{S}, \forall m \in \mathcal{M} \mid s' = s \oplus m\}$.

Des propriétés communes

- ▷ les ⊕ :
  - ◇ robustes (part d'aléatoire)
  - ◇ directes (pas de recours aux dérivées)
  - ◇ algorithmes simples et génériques
  - ◇ nombreuses extensions et hybrides
- ▷ les ⊖ :
  - ◇ méthodes approchées : pas de garantie que $s_f = s^\star$
  - ◇ lentes
  - ◇ paramètres difficiles à régler
  - ◇ choix difficile de la méthode la plus appropriée

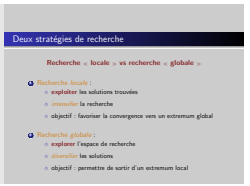


Page 9 :

Deux stratégies de recherche

Recherche ≪ locale ≫ vs recherche ≪ globale ≫

1. Recherche *locale* :
   - ◇ **exploiter** les solutions trouvées
   - ◇ *intensifier* la recherche
   - ◇ objectif : favoriser la convergence vers un extremum global
2. Recherche *globale* :
   - ◇ **explorer** l'espace de recherche
   - ◇ *diversifier* les solutions
   - ◇ objectif : permettre de sortir d'un extremum local



Page 10 :

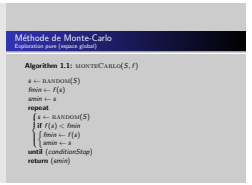
Deux stratégies opposées qu'il faut combiner efficacement...

Méthode de Monte-Carlo

Exploration pure (espace global)

Algorithm 1.1: MONTECARLO(S, f)

$s \leftarrow \text{RANDOM}(S)$
$fmin \leftarrow f(s)$
$smin \leftarrow s$
repeat
  $s \leftarrow \text{RANDOM}(S)$
  if $f(s) < fmin$
    $fmin \leftarrow f(s)$
    $smin \leftarrow s$
until $(conditionStop)$
return $(smin)$



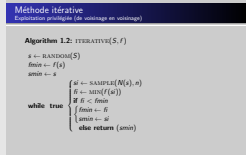
Page 11 :

Méthode itérative

Exploitation privilégiée (de voisinage en voisinage)

Algorithm 1.2: ITERATIVE(S, f)

$s \leftarrow \text{RANDOM}(S)$
$fmin \leftarrow f(s)$
$smin \leftarrow s$
while **true**
  $si \leftarrow \text{SAMPLE}(N(s), n)$
  $fi \leftarrow \text{MIN}(f(si))$
  if $fi < fmin$
    $fmin \leftarrow fi$
    $smin \leftarrow si$
  else return $(smin)$

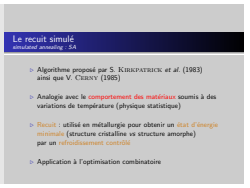


Page 12 :

Le recuit simulé

simulated annealing : SA

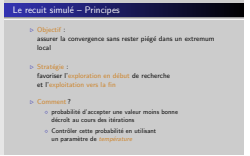
- Algorithme proposé par S. KIRKPATRICK *et al.* (1983) ainsi que V. CERNY (1985)
- Analogie avec le comportement des matériaux soumis à des variations de température (physique statistique)
- Recuit : utilisé en métallurgie pour obtenir un état d'énergie minimale (structure cristalline *vs* structure amorphe) par un refroidissement contrôlé
- Application à l'optimisation combinatoire



Page 13 :

Le recuit simulé – Principes

- Objectif :
  assurer la convergence sans rester piégé dans un extremum local
- Stratégie :
  favoriser l'exploration en début de recherche et l'exploitation vers la fin
- Comment ?
  - probabilité d'accepter une valeur moins bonne décroît au cours des itérations
  - Contrôler cette probabilité en utilisant un paramètre de *température*



Page 14 :

Résultats de la physique statistique

- Loi de Boltzmann :

  lorsqu'un équilibre thermodynamique est atteint à une température T, la probabilité de posséder une énergie E est :

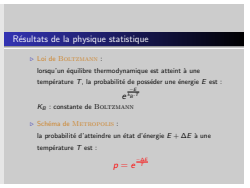
  $$e^{\frac{-E}{k_B \cdot T}}$$

  K_B : constante de Boltzmann

- Schéma de Metropolis :

  la probabilité d'atteindre un état d'énergie $E + \Delta E$ à une température T est :

  $$p = e^{\frac{-\Delta E}{T}}$$



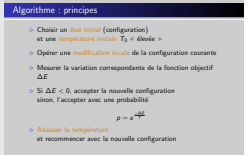
Page 15 :

Algorithme : principes

- Choisir un état initial (configuration)
  et une température initiale $T_0 \ll$ élevée $\gg$
- Opérer une modification locale de la configuration courante
- Mesurer la variation correspondante de la fonction objectif ΔE
- Si $\Delta E < 0$, accepter la nouvelle configuration
  sinon, l'accepter avec une probabilité

  $$p = e^{\frac{-\Delta E}{T}}$$

- Abaisser la température
  et recommencer avec la nouvelle configuration



Page 16 :

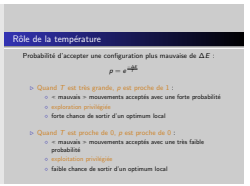
E est la fonction dont on cherche le minimum (*fonction de coût* ou *objectif*) Elle est calculable pour toutes les configurations s de l'espace de recherche $\mathcal{S}$. Elle représente donc le problème $\mathcal{P}$ que l'on cherche à résoudre.
La température T est un paramètre propre à l'algorithme. Elle n'a aucune signification pour le problème que l'on cherche à résoudre. Elle sert à guider la recherche.

Rôle de la température

Probabilité d'accepter une configuration plus mauvaise de ΔE :

$$p = e^{\frac{-\Delta E}{T}}$$

- Quand T est très grande, p est proche de 1 :
  - ≪ mauvais ≫ mouvements acceptés avec une forte probabilité
  - exploration privilégiée
  - forte chance de sortir d'un optimum local
- Quand T est proche de 0, p est proche de 0 :
  - ≪ mauvais ≫ mouvements acceptés avec une très faible probabilité
  - exploitation privilégiée
  - faible chance de sortir d'un optimum local

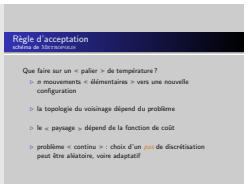


Page 17 :

Règle d'acceptation
schéma de METROPOLIS

Que faire sur un ≪ palier ≫ de température ?

- n mouvements ≪ élémentaires ≫ vers une nouvelle configuration
- la topologie du voisinage dépend du problème
- le ≪ paysage ≫ dépend de la fonction de coût
- problème ≪ continu ≫ : choix d'un *pas* de discrétisation peut être aléatoire, voire adaptatif



Page 18 :

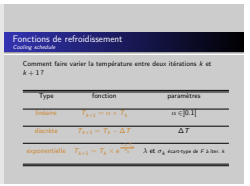
Les mouvements sont qualifiés d'élémentaires dans la mesure où ils ne doivent pas modifier fortement la configuration courante, d'une part parce que la recherche est locale et d'autre part qu'il faut pouvoir approcher (si possible trouver) l'optimum global.
Selon le type de problème que l'on veut résoudre, il est possible de définir l'*amplitude* d'un mouvement. On peut donc jouer sur ce paramètre pour améliorer la recherche. Dans le cas de l'optimisation d'une fonction continue, le choix d'une amplitude qui évolue au cours de la recherche est intéressante.

Fonctions de refroidissement

Cooling schedule

Comment faire varier la température entre deux itérations k et $k+1$?

Type	fonction	paramètres
linéaire	$T_{k+1} = \alpha \times T_k$	$\alpha \in]0.1[$
discrète	$T_{k+1} = T_k - \Delta T$	ΔT
exponentielle	$T_{k+1} = T_k \times e^{\frac{-\lambda . T_k}{\sigma_k}}$	λ et σ_k écart-type de F à iter. k

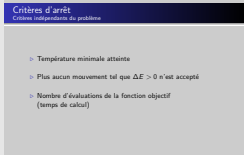


Page 19 :

Critères d'arrêt

Critères indépendants du problème

- Température minimale atteinte
- Plus aucun mouvement tel que $\Delta E > 0$ n'est accepté
- Nombre d'évaluations de la fonction objectif (temps de calcul)



Page 20 :

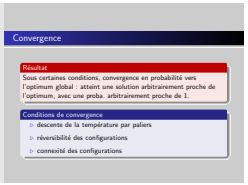
Convergence

Résultat

Sous certaines conditions, convergence en probabilité vers l'optimum global : atteint une solution arbitrairement proche de l'optimum, avec une proba. arbitrairement proche de 1.

Conditions de convergence

- ▷ descente de la température par paliers
- ▷ réversibilité des configurations
- ▷ connexité des configurations

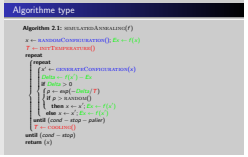


Page 21 :

Algorithme type

Algorithm 2.1: SIMULATEDANNEALING(f)

$x \leftarrow$ RANDOMCONFIGURATION(); $Ex \leftarrow f(x)$
$T \leftarrow$ INITTEMPERATURE()
repeat
  repeat
    $x' \leftarrow$ GENERATECONFIGURATION(x)
    $Delta \leftarrow f(x') - Ex$
    if $Delta > 0$
      $p \leftarrow exp(-Delta/T)$
      if $p >$ RANDOM()
        then $x \leftarrow x'$; $Ex \leftarrow f(x')$
      else $x \leftarrow x'$; $Ex \leftarrow f(x')$
  until $(cond - stop - palier)$
  $T \leftarrow$ COOLING()
until $(cond - stop)$
return (x)

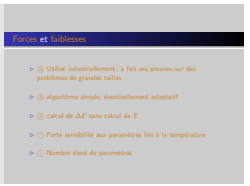


Page 22 :

REMARQUE : il existe plusieurs variantes de cet algorithme.

Forces et faiblesses

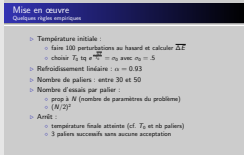
- ⊕ Utilisé industriellement ; a fait ses preuves sur des problèmes de grandes tailles
- ⊕ algorithme simple, éventuellement adaptatif
- ⊕ calcul de ΔE sans calcul de E
- ⊖ Forte sensibilité aux paramètres liés à la température
- ⊖ Nombre élevé de paramètres



Page 23 :

Mise en œuvre
Quelques règles empiriques

- Température initiale :
  - faire 100 perturbations au hasard et calculer $\overline{\Delta E}$
  - choisir T_0 tq $e^{\frac{-\overline{\Delta E}}{T_0}} = \sigma_0$ avec $\sigma_0 = .5$
- Refroidissement linéaire : $\alpha = 0.93$
- Nombre de paliers : entre 30 et 50
- Nombre d'essais par palier :
  - prop à N (nombre de paramètres du problème)
  - $(N/2)^2$
- Arrêt :
  - température finale atteinte (cf. T_0 et nb paliers)
  - 3 paliers successifs sans aucune acceptation

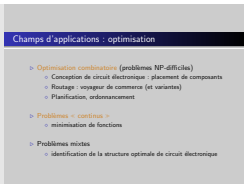


Page 24 :

Ces règles ne doivent pas être prises au pied de la lettre ; elles constituent un point de départ dans la recherche d'un bon réglage de la méthode. Toutes ces valeurs ne sont d'ailleurs pas compatibles...

Champs d'applications : optimisation

- Optimisation combinatoire (problèmes NP-difficiles)
  - Conception de circuit électronique : placement de composants
  - Routage : voyageur de commerce (et variantes)
  - Planification, ordonnancement
- Problèmes « continus »
  - minimisation de fonctions
- Problèmes mixtes
  - identification de la structure optimale de circuit électronique



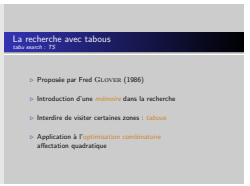
Page 25 :

L'identification de la structure optimale de circuit électronique consiste à trouver automatiquement un modèle de circuit électronique (linéaire) avec le moins possible de composants et reproduisant au mieux un comportement donné (à partir d'observations expérimentales).

La recherche avec tabous

tabu search : TS

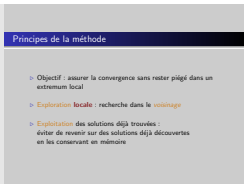
- Proposée par Fred Glover (1986)
- Introduction d'une *mémoire* dans la recherche
- Interdire de visiter certaines zones : *tabous*
- Application à l'optimisation combinatoire
  affectation quadratique



Page 26 :

Principes de la méthode

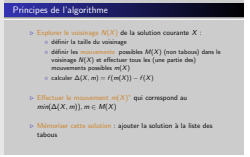
- Objectif : assurer la convergence sans rester piégé dans un extremum local
- Exploration **locale** : recherche dans le *voisinage*
- Exploitation des solutions déjà trouvées :
  éviter de revenir sur des solutions déjà découvertes
  en les conservant en mémoire



Page 27 :

Principes de l'algorithme

- Explorer le voisinage $N(X)$ de la solution courante X :
  - définir la taille du voisinage
  - définir les *mouvements* possibles $M(X)$ (non tabous) dans le voisinage $N(X)$ et effectuer tous les (une partie des) mouvements possibles $m(X)$
  - calculer $\Delta(X, m) = f(m(X)) - f(X)$
- Effectuer le mouvement $m(X)^*$ qui correspond au $min(\Delta(X, m)), m \in M(X)$
- Mémoriser cette solution : ajouter la solution à la liste des tabous

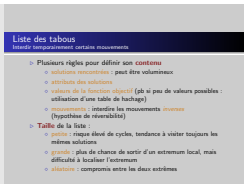


Page 28 :

Liste des tabous

Interdir temporairement certains mouvements

- Plusieurs règles pour définir son **contenu**
  - solutions rencontrées : peut être volumineux
  - attributs des solutions
  - valeurs de la fonction objectif (pb si peu de valeurs possibles : utilisation d'une table de hachage)
  - mouvements : interdire les mouvements *inverses* (hypothèse de réversibilité)
- **Taille** de la liste :
  - petite : risque élevé de cycles, tendance à visiter toujours les mêmes solutions
  - grande : plus de chance de sortir d'un extremum local, mais difficulté à localiser l'extremum
  - aléatoire : compromis entre les deux extrêmes



Page 29 :

Critère d'aspiration

Lever le tabou

- Autoriser un mouvement qui fait partie de la liste des tabous et qui améliorerait la solution courante (cas où les mouvements sont tabous)
- Effectuer un mouvement qui n'a pas été réalisé depuis longtemps (quelque soit son résultat)

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 30 / 58

Ajout d'heuristiques de mémorisation

- Stratégie d'≪ intensification ≫ : on mémorise les meilleures solutions et on essaie de dégager des propriétés communes pour définir des régions intéressantes
  (p. ex. en rendant tabou tous les mouvements qui font sortir de cette région)

- Stratégie de ≪ diversification ≫ : c'est le contraire, on mémorise les solutions les plus fréquemment visitées et on impose un système de pénalités pour favoriser les mouvements les moins souvent utilisés

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 31 / 58

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

Résultat théorique : convergence

Résultats théoriques partiels...
dans des conditions très restricives

... de faible utilité pratique...

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 32 / 58

Forces et faiblesses

- ⊕ Méthode pour l'optimisation combinatoire (problèmes discrets)
- ⊕ Algorithme simple (avec nombreuses variantes)
- ⊖ Soin à apporter à la définition du voisinage et à la nature des mouvements
- ⊖ Méthode assez subtile ; à maîtriser...

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 33 / 58

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

Applications

Optimisation combinatoire (problèmes discrets)

- Voyageur de commerce (TSP)
- Affectation quadratique : p. ex. placement de composants
- Tournées de véhicules : livrer n commandes de taille v_i en utilisant m véhicules de capacité V_j

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 34 / 58

Algorithmes de Colonies de Fourmis
Ant Colony Optimisation

- Méthode inventée par Dorigo (1992)
- Auto-organisation chez les insectes sociaux
- Recherche de nourriture chez les fourmis : problème complexe dans un environnement dynamique
- Optimisation difficile de problèmes combinatoires
- Nombreuses variantes (discret et continu)

Page 35 :

Dorigo est directeur de recherche au laboratoire IRIDIA de l'université Libre de Bruxelles. Il est l'inventeur de la métha-heuristique appelée *Ant Colony Optimization*
L'algorithme s'inspire du comportement de recherche de nourriture de certaines espèces de fourmis (*foraging behavior*). Lorsqu'une fourmi recherche de la nourriture, elle explore ≪ aléatoirement ≫ les environs du nid. Quand elle trouve de la nourriture qui lui convient, elle en ramène une portion au nid. Sur le trajet de retour, la fourmi dépose sur le sol une substance chimique volatile, appelée *phéromone*. L'existence de phéromone guide ensuite la recherche de nourriture des autres fourmis, réduisant ainsi le trajet parcouru.

ACO : auto-organisation et stigmergie
Ant Colony Optimisation – Principes

1. Interactions multiples entre individus : liberté individuelle et ≪ contraintes ≫ sociales
2. Amplification par rétro-action positive : mise à jour d'informations peu apparentes
3. Rétro-action négative : stabilisation du système (*homéostasie*)
4. **L'œuvre stimule l'action** : coordination d'actions par communication via l'environnement : *stigmergie*, P.-P. GRASSÉ, 1959.

Page 36 :

ACO : communication par phéromones

Ant Colony Optimisation – Principes

Dépot de marques, *phéromones*, dans l'environnement formant des *pistes* volatiles

Perception locale par les fourmis

Activation de comportement

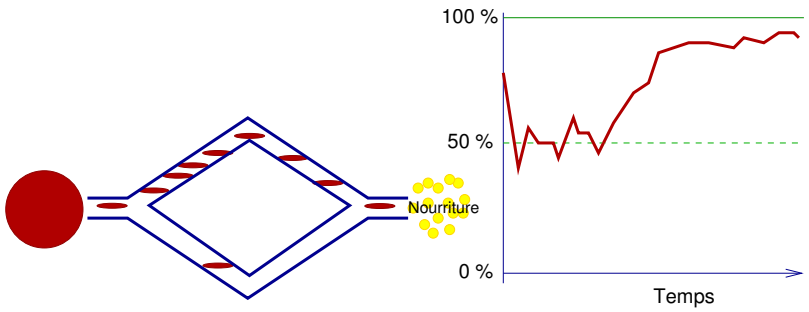
Evaporation des marques

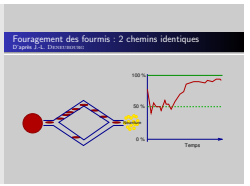
Déterministe *faible*

Page 37 :

Fouragement des fourmis : 2 chemins identiques

D'après J.-L. Deneubourg



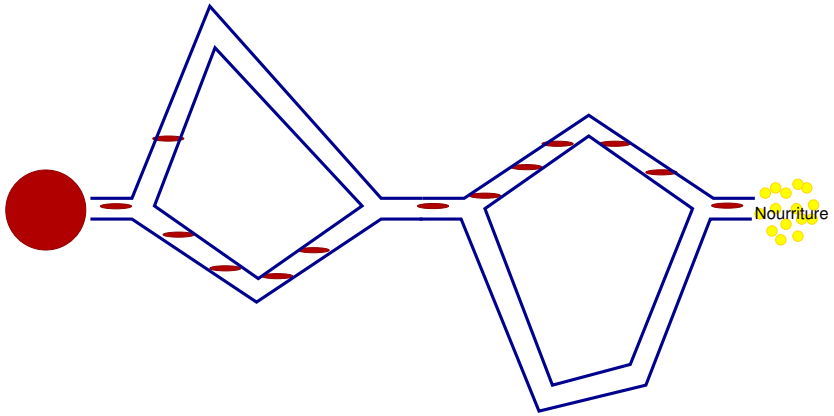


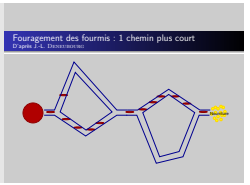
Page 38 :

Une petite différence entre les deux pistes se trouvent rapidement amplifiée.

Fouragement des fourmis : 1 chemin plus court

D'après J.-L. Deneubourg



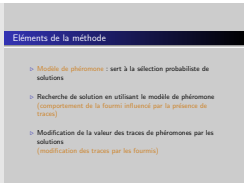


Page 39 :

Les fourmis qui empruntent la piste la plus courte reviennent au nid le plus vite ; la quantité de phéromone de cette piste est supérieure à l'autre, ce qui accroît la probabilité qu'elle soit choisie par les autres fourmis.

Eléments de la méthode

- Modèle de phéromone : sert à la sélection probabiliste de solutions
- Recherche de solution en utilisant le modèle de phéromone (comportement de la fourmi influencé par la présence de traces)
- Modification de la valeur des traces de phéromones par les solutions (modification des traces par les fourmis)



Page 40 :

ACO et voyageur de commerce

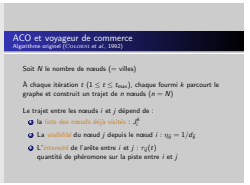
Algorithme originel (Colorni *et al.*, 1992)

Soit N le nombre de nœuds (= villes)

À chaque itération t ($1 \leq t \leq t_{\max}$), chaque fourmi k parcourt le graphe et construit un trajet de n nœuds ($n = N$)

Le trajet entre les nœuds i et j dépend de :

1. la liste des nœuds déjà visités : J_i^k
2. La *visibilité* du nœud j depuis le nœud i : $\eta_{ij} = 1/d_{ij}$
3. L'*intensité* de l'arête entre i et j : $\tau_{ij}(t)$
   quantité de phéromone sur la piste entre i et j

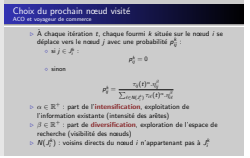


Page 41 :

Choix du prochain nœud visité

ACO et voyageur de commerce

- À chaque itération t, chaque fourmi k située sur le nœud i se déplace vers le nœud j avec une probabilité p_{ij}^k :
  - si $j \in J_i^k$:
    $$p_{ij}^k = 0$$
  - sinon
    $$p_{ij}^k = \frac{\tau_{ij}(t)^\alpha . \eta_{ij}^\beta}{\sum_{\ell \in N(J_i^k)} \tau_{i\ell}(t)^\alpha . \eta_{i\ell}^\beta}$$
- $\alpha \in \mathbb{R}^+$: part de l'**intensification**, exploitation de l'information existante (intensité des arêtes)
- $\beta \in \mathbb{R}^+$: part de **diversification**, exploration de l'espace de recherche (visibilité des nœuds)
- $N(J_i^k)$: voisins directs du nœud i n'appartenant pas à J_i^k

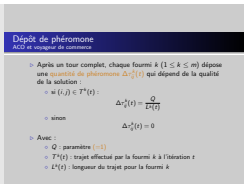


Page 42 :

Dépôt de phéromone

ACO et voyageur de commerce

- Après un tour complet, chaque fourmi k $(1 \leq k \leq m)$ dépose une quantité de phéromone $\Delta\tau_{ij}^k(t)$ qui dépend de la qualité de la solution :
  - si $(i,j) \in T^k(t)$:
$$\Delta\tau_{ij}^k(t) = \frac{Q}{L^k(t)}$$
  - sinon
$$\Delta\tau_{ij}^k(t) = 0$$
- Avec :
  - Q : paramètre $(=1)$
  - $T^k(t)$: trajet effectué par la fourmi k à l'itération t
  - $L^k(t)$: longueur du trajet pour la fourmi k



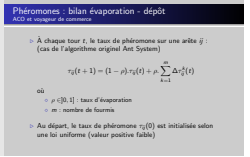
Page 43 :

Source : M. Dorigo *et al.*, 2000.

Phéromones : bilan évaporation - dépôt

ACO et voyageur de commerce

- À chaque tour t, le taux de phéromone sur une arête ij : (cas de l'algorithme originel Ant System)
$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \rho.\sum_{k=1}^{m}\Delta\tau_{ij}^k(t)$$
où
  - $\rho \in]0,1]$: taux d'évaporation
  - m : nombre de fourmis
- Au départ, le taux de phéromone $\tau_{ij}(0)$ est initialisée selon une loi uniforme (valeur positive faible)



Page 44 :

Il est possible de remplacer le terme $\Delta\tau_{ij}^k(t)$ par une *fonction de fitness*, ce qui permet d'appliquer l'algorithme à d'autres problèmes.

Applications

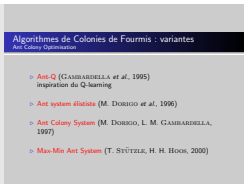
- Optimisation combinatoire
  - Voyageur de commerce (et variantes)
  - Affectation quadratique
  - Planification, ordonnancement
  - Routage de véhicules
- Problème continu
  - minimisation de fonctions
- Problème mixte
  - identification de la structure optimale de circuit électronique

Quelques types de problème d'optimisation combinatoire difficile :

- TSP : *Traveling Salesman Problem*, voyageur de commerce ;
- ATSP : *Asymmetric Traveling Salesman Problem* : voyageur de commerce dans le cas où le coût de A vers B diffère de celui de B vers A ;
- SOP : *Sequential Ordering Problem*, variante de l'ATSP avec des contraintes de précédence sur les arêtes du graphe.

Algorithmes de Colonies de Fourmis : variantes
Ant Colony Optimisation

- Ant-Q (Gambardella *et al.*, 1995)
  inspiration du Q-learning
- Ant system élististe (M. Dorigo *et al.*, 1996)
- Ant Colony System (M. Dorigo, L. M. Gambardella, 1997)
- Max-Min Ant System (T. Stützle, H. H. Hoos, 2000)



Page 46 :

Ant system élististe (DORIGO *et al.*, 1996)

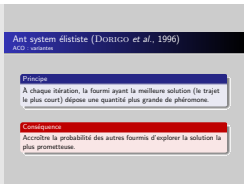
ACO : variantes

Principe

À chaque itération, la fourmi ayant la meilleure solution (le trajet le plus court) dépose une quantité plus grande de phéromone.

Conséquence

Accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse.



Page 47 :

ACS : Ant Colony System (DORIGO *et al.*, 1997)

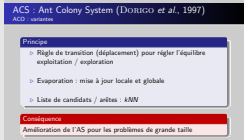
ACO : variantes

Principe

- ▷ Règle de transition (déplacement) pour régler l'équilibre exploitation / exploration
- ▷ Evaporation : mise à jour locale et globale
- ▷ Liste de candidats / arêtes : *kNN*

Conséquence

Amélioration de l'AS pour les problèmes de grande taille



Page 48 :

Marco Dorigo & Luca Maria Gambardella, 1997, Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation, Vol.1, No.1.

MMAS : Max-Min Ant System (Stützle *et al.*, 1997)

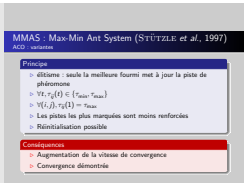
ACO : variantes

Principe

- élitisme : seule la meilleure fourmi met à jour la piste de phéromone
- $\forall t, \tau_{ij}(t) \in \{\tau_{\min}, \tau_{\max}\}$
- $\forall (i,j), \tau_{ij}(1) = \tau_{\max}$
- Les pistes les plus marquées sont moins renforcées
- Réinitialisation possible

Conséquences

- Augmentation de la vitesse de convergence
- Convergence démontrée



Page 49 :

Les Algorithmes de Colonies de Fourmis : applications

Ant Colony Optimisation

Affectation quadratique

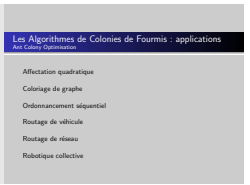
Coloriage de graphe

Ordonnancement séquentiel

Routage de véhicule

Routage de réseau

Robotique collective

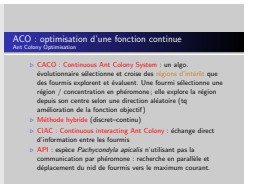


Page 50 :

ACO : optimisation d'une fonction continue

Ant Colony Optimisation

- CACO : Continuous Ant Colony System : un algo. évolutionnaire sélectionne et croise des régions d'intérêt que des fourmis explorent et évaluent. Une fourmi sélectionne une région / concentration en phéromone ; elle explore la région depuis son centre selon une direction aléatoire (tq amélioration de la fonction objectif)
- Méthode hybride (discret–continu)
- CIAC : Continuous interacting Ant Colony : échange direct d'information entre les fourmis
- API : espèce *Pachycondyla apicalis* n'utilisant pas la communication par phéromone : recherche en parallèle et déplacement du nid de fourmis vers le maximum courant.



Page 51 :

Auto-organisation chez les animaux sociaux

- De nombreux animaux ont des comportements collectifs de déplacement
  - essaims d'insectes (criquets...)
  - nuées d'oiseaux (étourneaux...)
  - bancs de poissons (sardines...)
  - troupeaux d'herbivores (gnous...)
- Le collectif se déplace très vite
- Il peut se scinder à l'approche d'un obstacle et se reformer ensuite
- Il peut être attiré par un objet (un arbre...) puis « sauter » ensuite à un autre

Les boids de C. G. Reynolds

Émergence à partir d'interactions simples

- Règle 1 : **attraction**
  propention, pour un individu isolé, à rejoindre un groupe
- Règle 2 : **cohésion**
  tendance à adopter une vitesse et une orientation identique à celle de ses voisins immédiats
- Règle 3 : **répulsion**
  maintien d'une distance inter-individuelle minimale

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

L'optimisation par essaim de particules (R. Eberhart & J. Kennedy, 1995)

PSO : Particule Swarm Optimisation

- À chaque particule i, on associe :
  - sa position $\vec{x}_i \in \mathbb{R}^n$
  - sa vélocité $\vec{v}_i \in \mathbb{R}^n, \vec{v}_i < \vec{v}_{\max}$
  - sa mémoire de sa dernière meilleure position $\vec{p}_i$
- À chaque itération, la particule doit se déplacer de façon à se rapprocher de l'optimum recherché (max.)

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

L'ouvrage de référence : J. Kennedy, R. C. Eberhart, 2001, *Swarm intelligence*, Morgan Kaufmann.

Comportement d'une particule

Compromis entre 3 tendances :

1. suivre sa propre voie (conserver sa vitesse)
2. revenir à ce qu'elle connait (tendance conservatrice)
3. suivre les autres

Pour cela, on combine 3 informations :

1. sa vitesse $\vec{v}_i(t)$
2. sa meilleure performance $\vec{p}_i$
3. la meilleure performance des voisines $\vec{p}_g$

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 55 / 58

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

Algorithme originel

À chaque itération t,

la particule i se déplace de :

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$$

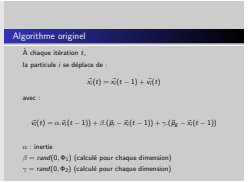
avec :

$$\vec{v}_i(t) = \alpha.\vec{v}_i(t-1)) + \beta.(\vec{p}_i - \vec{x}_i(t-1)) + \gamma.(\vec{p}_g - \vec{x}_i(t-1))$$

α : inertie

$\beta = rand(0, \Phi_1)$ (calculé pour chaque dimension)

$\gamma = rand(0, \Phi_2)$ (calculé pour chaque dimension)



Page 56 :

The three terms in the velocity-update rule above characterize the local behaviors that particles follow. The first term, called the inertia or momentum serves as a memory of the previous flight direction, preventing the particle from drastically changing direction. The second term, called the cognitive component models the tendency of particles to return to previously found best positions. The third term, called the social component quantifies the performance of a particle relative to its neighbors. It represents a group norm or standard that should be attained.

Paramètres de l'algorithme

Nombre de particules

Définition du voisinage

Coefficients de pondération

Vitesse maximale

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 57 / 58

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

Types d'application

Problèmes continus : optimisation de fonction

Concurrent des algorithmes génétiques

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références
Présentation de la méthode
Analyse de la méthode
Application de la méthode

C.B. Méta-heuristiques 58 / 58

Dréo, J., Pétrowsky, A., Patrick, S., and Taillard, E. (2003).
Métaheuristiques pour l'optimisation difficile.
Eyrolles.

Glover, F. and Laguna, M. (1997).
Tabu Search.
Kluwer Academic Publishers, 2nd edition edition.

Kennedy, J. and Eberhart, R. C. (2002).
Swarm Intelligence.
Morgan Kaufmann Publishers.

Monmarché, N., Guinand, F., and Siarry, P., editors (2009a).
Fourmis artificielles 1 - des bases de l'optimisation aux applications industrielles.
Hermes Science - Lavoisier.

Monmarché, N., Guinand, F., and Siarry, P., editors (2009b).
Fourmis artificielles 2 - nouvelles directions pour une intelligence collective.

Introduction
Recuit simulé
Recherche avec tabous
Algorithmes de colonies de fourmis
Optimisation par essaim de particules
Références

Hermes Science - Lavoisier.

C.B. Méta-heuristiques 58 / 58