Cours de Master Recherche
Spécialité CODE :
Résolution de problèmes combinatoires

Christine Solnon

LIRIS, UMR 5205 CNRS / Université Lyon 1

2008

Rappel du plan du cours

16 heures de cours

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

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

Résolution du problème SAT par recherche de cliques

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

Approches 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

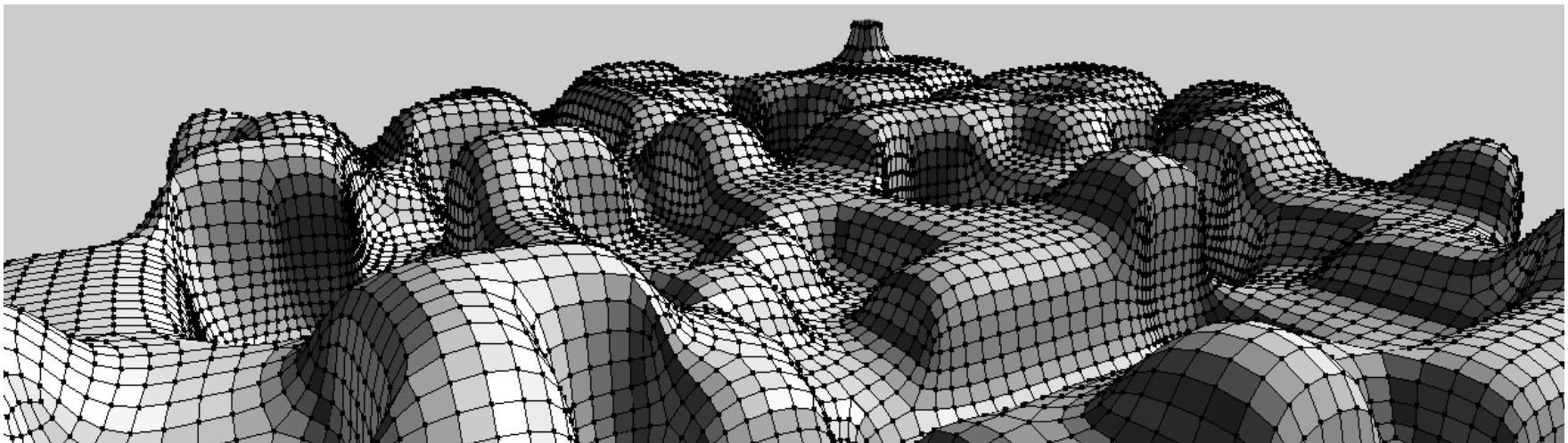
Définition d'un paysage de recherche :

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



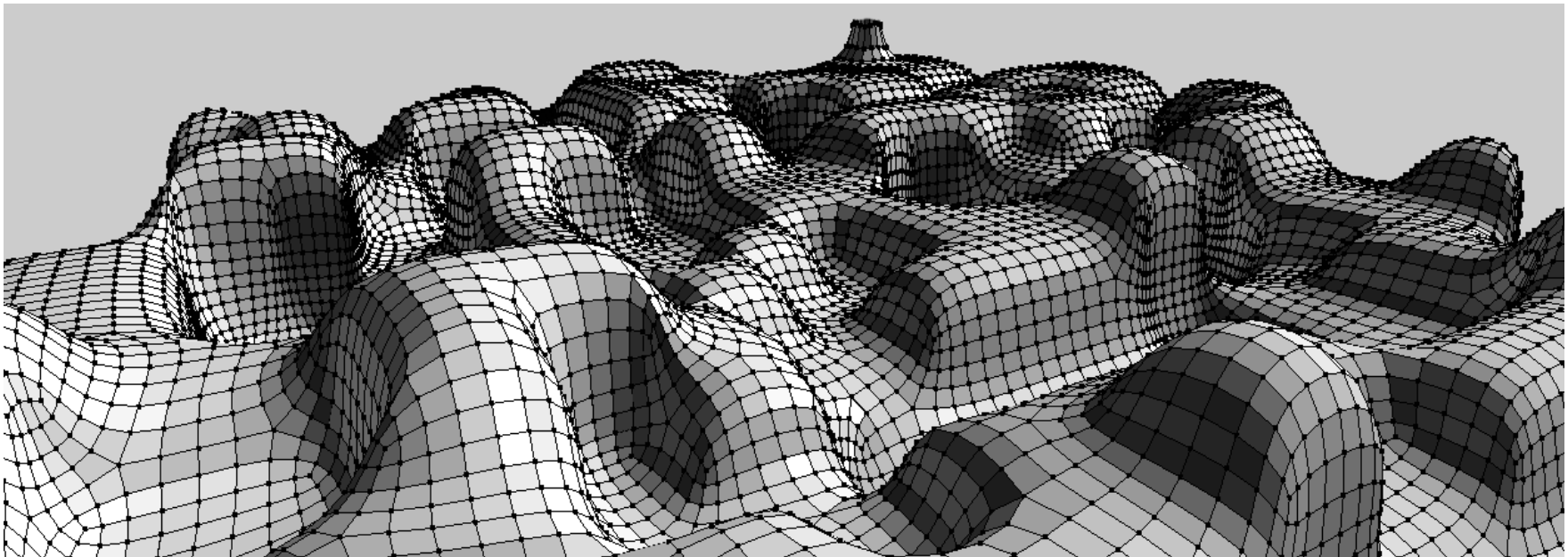
Optima locaux et plateaux (1)

Définitions

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

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

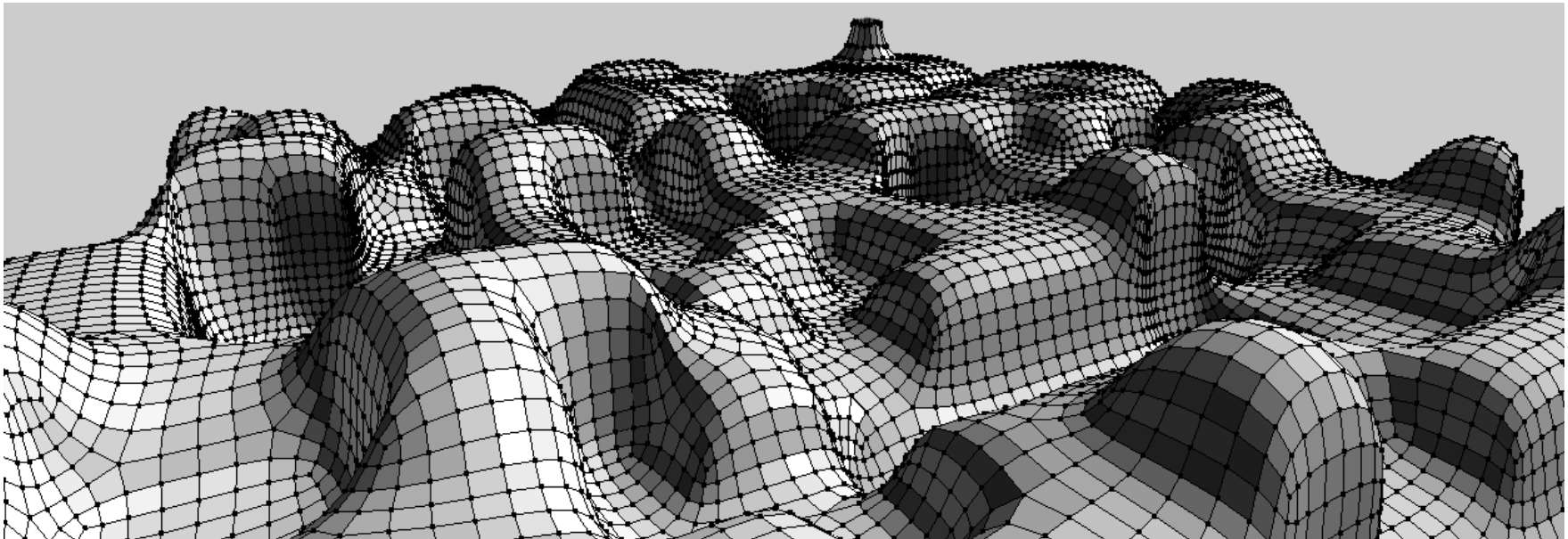
- Plateau = ensemble de points connexes dans le graphe $G = (E, V)$ ayant tous la même valeur pour f
- Bassin d'attraction d'un optimum local e
  $\rightsquigarrow$ points que l'on peut atteindre depuis e sans jamais monter



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