INCOP 1.0 : mode d'emploi

Bertrand Neveu

17 septembre 2003

1 Problèmes traités et méthodes proposées

Cette bibliothèque logicielle traite des problèmes d'optimisation combinatoire par des méthodes incomplètes à base de recherche locale. Tous les problèmes sont des problèmes de minimisation mono-critère. Ce mode d'emploi indique comme utiliser la bibliothèque dans son état actuel (version 1.0) : chaque problème implanté a un exécutable qui est appelé par une ligne de commande. Toutes les méthodes sont appelables dans chaque exécutable. Les différents paramètres à entrer pour les problèmes et les méthodes sont indiqués ci-après.

Un fichier makefile indique comment construire les exécutables sous linux à partir des fichiers sources.

Pour une présentation générale de la bibliothèque, se référer au document de présentation.

1.1 Problèmes traités

Ce programme traite actuellement les CSP binaires donnés en extension (CSP aléatoires, CSP au format ds), le problème des n-reines, le coloriage de graphe (modélisation max-CSP ou modélisation impasse), le problème de clique maximum et les problèmes d'allocation de fréquences (celar6, celar7 et celar8).

1.2 Méthodes proposées

Des méthodes suivantes de recherche locale et de recherche à population (gww : go with the winners) et des méthodes hybrides (gww+ recherche locale) sont proposées dans la bibliothèque.

Les métaheuristiques des méthodes de recherche locale suivantes sont implantées dans la bibliothèque :

1. metropolis ; une marche aléatoire avec critère de Metropolis
2. recuit simulé : baisse linéaire de la température
3. liste taboue : une marche aléatoire avec liste taboue
4. acceptation avec seuil (threshold accepting) : les mouvements détériorants sont acceptés si le delta est inférieur à un seuil qui baisse linéairement entre une valeur max et 0.

5. descente : seuls les mouvements améliorant l'évaluation ou la laissant constante sont acceptés.
6. marche aléatoire : tout mouvement est accepté (n'a d'intérêt que comme marche de diversification de gww)

Les méthodes à population suivantes sont implantées :

1. gww (avec seuil et marche aléatoire ou recherche locale) : la marche aléatoire est limitée par le seuil. Pour la marche aléatoire, on a le choix entre toutes les méthodes de recherche locale implantées dans la bibliothèque (descente, Metropolis, liste taboue, ...) Il y a 5 variantes pour la gestion de la baisse du seuil.
   - gww : descente du seuil par un taux jusqu'à une valeur minimale.
   - gww-killworst : descente du seuil avec un taux en partant de la plus mauvaise particule.
   - gww-adapt : descente du seuil pour éliminer K mauvaises particules
   - gww-mediandist : descente déterminée par un taux de la distance du pire au médian
   - gww-bestdist : descente déterminée par un taux de la distance du pire au meilleur
2. gww-nothreshold (pas de seuil et marche aléatoire ou recherche locale)

2 Fichier de sortie

Le premier paramètre est le nom du fichier de sortie où sont les résultats et la trace. Si ce paramètre vaut la chaîne `arg`, le fichier a pour nom la concaténation des arguments de la ligne de commandes dans un sous-répertoire de nom `results` à créer avant de lancer le programme.

3 Paramètres d'appel pour les problèmes

3.1 Coloriage de graphes

Les graphes doivent être dans le format standard Dimacs.
Exécutable : colorcsp ou impassecolor

Voici la liste des paramètres :
- **fichier** : chaîne : nom du fichier du graphe au format Dimacs
- **nb-couleurs** : nombre de couleurs
  en cas d'utilisation de la modélisation impasse , 1 paramètre supplémentaire est à fournir :
  - **critère** : 0 pour le nombre de variables non affectées, 1 pour la somme des degrés des variables non affectées

3.2 Clique maximum

Il s'agit de chercher dans un graphe une clique de taille n donnée. Comme pour le coloriage, les graphes doivent être dans le format Dimacs.
Exécutable : maxclique

Le voisinage est un voisinage de type "échange".

Voici la liste des paramètres :

- **fichier** : nom du fichier du graphe au format Dimacs
- **taille-clique** : taille de la clique recherchée.

3.3 CSP aléatoires

Les CSP aléatoires sont générés à l'aide du générateur urbcsp de Christian Bessière.
Exécutable : randomcsp

Voici la liste des paramètres :

- **type-problème** : random pour indiquer qu'il s'agit d'un CSP aléatoire
- **nb-variables** : nombre de variables
- **taille-domaines** : taille des domaines
- **nb-contraintes** : nombre de contraintes
- **nb-couples** : nombre de couples interdits par contrainte
- **graine-générateur** : graine du générateur aléatoire
- **nb-problèmes** : nombre de problèmes traités

3.4 CELAR

Les 3 problèmes du CELAR d'optimisation d'une somme pondérée des contraintes violées (6 , 7 et 8) sont traités.
Exécutable : celarincr (mode semi-incrémental) ou celarfullincr (mode incrémental)

- **type-problème** : celar
- **fichier** : celar6.txt, celar7.txt ou celar8.txt (fichier regroupant les données de chacun des problèmes)

3.5 N-reines

Le problème test de placer N-reines sur un échiquier de taille N*N, sans qu'elles soient en prise.
Exécutable : queen16 (implantation la plus efficace : mouvement d'échange avec évaluation non incrémentale)

- **nb-reines** : entier : nombre de reines

3.6 CSP binaires en extension

Le problème est un CSP binaire en extension traité en Max-CSP valué (somme pondérée des violations de contraintes). Format du fichier d'entrée (ds

de Simon de Givry).
Exécutable : wextcsp

- **nom-fichier** : nom du fichier

4 Paramètres d'appel pour les méthodes

Deux grands types de méthodes sont implantés : des méthodes de recherche locale (tabou, recuit simulé...) et des méthodes à population (gww) . Pour la méthode à population gww, on doit préciser une méthode de recherche locale ou de marche aléatoire.

- **graine** : entier : graine pour les tirages aléatoires
- **nb-essais** : entier : nombre d'essais réalisés sur un problème (en incrémentant la graine des tirages aléatoires à chaque essai)
- **méthode** : chaîne : méthode utilisée (recherche locale : tabu, metropolis,simann,threshold, random, greedy, méthode à population gww gww-killworst,gww-adapt, gww-mediandist, gww-bestdist, gww-nothreshold)
  Pour les méthodes de recherche locale, les paramètres sont (nb mouvements, paramètres de la métaheuristique, paramètres du voisinage)
- **nb-mouvements** : entier : nombre total de mouvements

  Selon la métaheuristique utilisée, les paramètres suivants sont à entrer :
  metropolis : algorithme de Metropolis
  - **temperature** : flottant : température constante pour l'acceptation d'un mouvement.

  tabu
  - **longueur-liste** : entier : longueur de la liste taboue

  threshold : acceptation avec seuil
  - **seuil départ** : entier : delta maximum autorisé au départ

  simann : recuit simulé
  - **température initiale** : flottant : température initiale (baisse linéaire au long de l'algorithme)

  Paramètres voisinage
  - **var-conflit** : chaîne : toute variable (v) ou variable participant à un conflit (cv)
  - **min-conflit** : chaîne : valeur minimisant les conflits (mc) ou au hasard (v)
  - **min-voisins** : entier : nombre minimum de voisins explorés
  - **max-voisins** : entier : nombre maximum de voisins explorés
  - **épuisement** : entier : que faire quand le voisinage est épuisé (0 : rien, 1 : premier voisin sous le seuil, 2 : meilleur voisin)

  Pour les méthodes à population, **gww ou gww-killworst ou gww-adapt ou gww-mediandist ou gww-bestdist ou gww-nothreshold**, les paramètres suivants sont à fournir :
  - **nb-particules** entier : nombre de particules
  - **test-regroupement** booléen : marche seulement si regroupement au-

paravant
- **baisse dernier mouvement** booléen : baisse du seuil au dernier pas de la marche
- **élitisme** : booléen : on remet dans la population le meilleur trouvé à chaque regroupement
- **arrêt-marche** : booléen : on arrête la marche d'une particule dès qu'on n'arrive pas à trouver de mouvement faisable sous le seuil (pas de stagnation).
- Paramètres pour gww : baisse régulière du seuil et
  pour gww-killworst : baisse avec taux constant à partir de la plus mauvaise particule
  - **taux baisse seuil** flottant : (exemple .01 si on baisse le seuil de 1%)
  - **borne inférieure** entier : gww : nouveau seuil = (1 - taux baisse seuil) (ancien-seuil - borne inférieure)
    gww-killworst : nouveau seuil = (1 - taux baisse seuil) (pire - borne inférieure)
- Paramètres pour gww-adapt
  - **nb-tues** entier : nombre de particules au moins éliminées à chaque itération (sert à déterminer la baisse du seuil)
- Paramètre pour gww-mediandist
  - **distance-median** flottant : taux de la distance du pire au médian utilisé pour placer la nouvelle valeur du seuil
- Paramètre pour gww-bestdist
  - **distance-meilleur** flottant : taux de la distance du pire au meilleur utilisé pour placer la nouvelle valeur du seuil
- Paramètres pour gww-nothreshold
  - **nb-tués** entier : nombre de particules éliminées à chaque itération
  - **nb-itérations** entier : nombre d'itérations

- **type-marche** chaîne : métaheuristique pour la recherche locale utilisée : chaîne parmi les métaheuristiques implantées : tabu,metropolis,random (marche aléatoire), greedy (descente), simann (recuit simulé), threshold (acceptation à seuil)
  Paramètres marche, métaheuristique et voisinage : cf ci-dessus.

5 Paramètres de trace

Mode trace : 0 si pas de trace, 1 trace à chaque meilleur trouvé et à chaque changement de seuil pour les méthodes gww avec seuil.

6 Exemples

Exemples de lignes de commande :
colorcsp resqueen queen8_8.col 8 0 1 gww 100 0 1 1 0 0.05 200 metropolis 100 0.4 cv v 0 30 2 1

celarcsp rescelar8 celar8.txt 0 1 gww 100 0 1 1 0 0.05 320 metropolis 100 0.4 cv v 0 176 2 1
celarcsp rescelar8 celar8.txt 0 1 gww-adapt 100 0 1 1 0 5 metropolis 100 0.4 cv v 0 176 2 1
celarcsp rescelar8 celar8.txt 0 1 metropolis 1000000 .4 cv mc 0 176 2 1
celarcsp arg celar8.txt 0 1 tabu 1000000 20 v v 20 176 2 0
randomcsp resrcsp random 100 15 1000 52 0 10 0 4 metropolis 3000000 0.15 v v 0 800 1 1
randomcsp resrcsp random 100 15 1000 52 0 10 0 4 random 3000000 cv v 0 800 1 1