Recherche local et mémétique : de la théorie à la pratique

Jin-Kao Hao

LERIA
Université d'Angers
2, Boulevard Lavoisier
F- 49045 Angers Cedex 01

téléphone : 02 41 73 50 76
email : hao@info.univ-angers.fr
Web : www.info.univ-angers.fr/pub/hao

PLAN

1. Optimisation combinatoire
2. Panorama sur les méthodes de résolution
3. Approche de recherche locale
4. Approche d'évolution
5. Approche d'hybridation
6. Exemple d'applications
7. Conclusions

1ère partie

Panorama sur les méthodes de résolution

Cette partie est fondée sur l'article de synthèse suivante :

J.K. Hao, P. Galinier et M. Habib
Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes.
Revue d'Intelligence Artificielle Vol.13 (2) : 283-324, 1999.

Article accessible à : www.info.univ-angers.fr/pub/hao

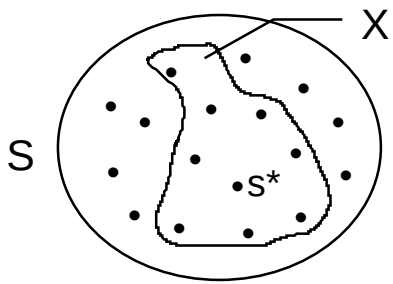
Optimisation combinatoire

Minimisation

étant donné un couple (S,f) où

- S un ensemble fini de solutions ou configurations (espace de recherche)
- f : S$\rightarrow$R une fonction de coût (ou objectif)

trouver $s^* \in X \subseteq S$ tel que $f(s^*) \leq f(s)\ \forall\ s \in X$ (config. faisables ou réalisables)

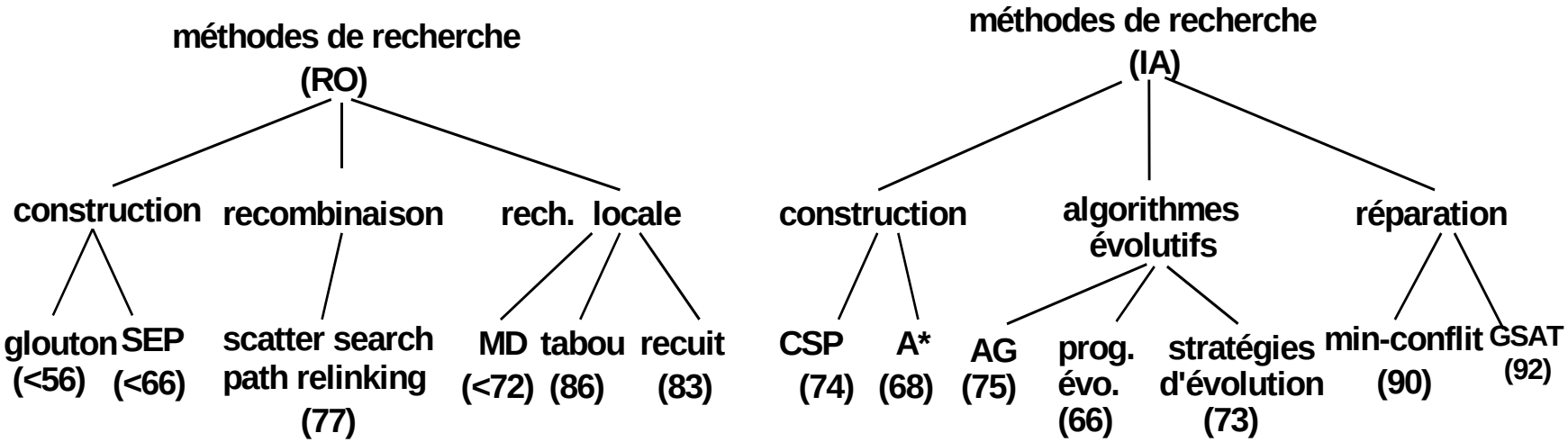


Exemple : voyager de commerce (TSP)

Remarques :

- pour la maximisation, il suffit de remplacer "f(s*) $\leq$ f(s)" par " f(s*) $\geq$ f(s)"
- en pratique, S et f ne sont pas nécessairement donnés à l'avance (pb codage)
- la plupart des pb d'optimisation intéressants sont dans la classe NP-difficile

Méthodes de résolution



Quatre grandes approches :

1. **Construction** : instanciation successive des variables selon un ordre statique ou dynamique (branch & bound, CSP, méthodes gloutonnes...)
2. **Recherche locale** : réparation itérative d'une configuration complète par des modifications locales (descente, recuit simulé, recherche tabou, min-conflit...)
3. **Evolution** : évolution d'une population de solutions par des opérations "génétiques" (sélection, croisement, et mutation), ex : algorithmes génétiques...
4. **Hybridation** : combinaison de différentes approches (évolution + RL, évolution + construction...)

Méthodes de résolution

Approche de construction

- l'instanciation successive des variables selon un ordre statique ou dynamique.
- si exacte (complète), alors complexité exponentielle dans le pire des cas
- ex : branch & bound, CSP, méthodes gloutonnes...

Approche de recherche locale ou voisinage

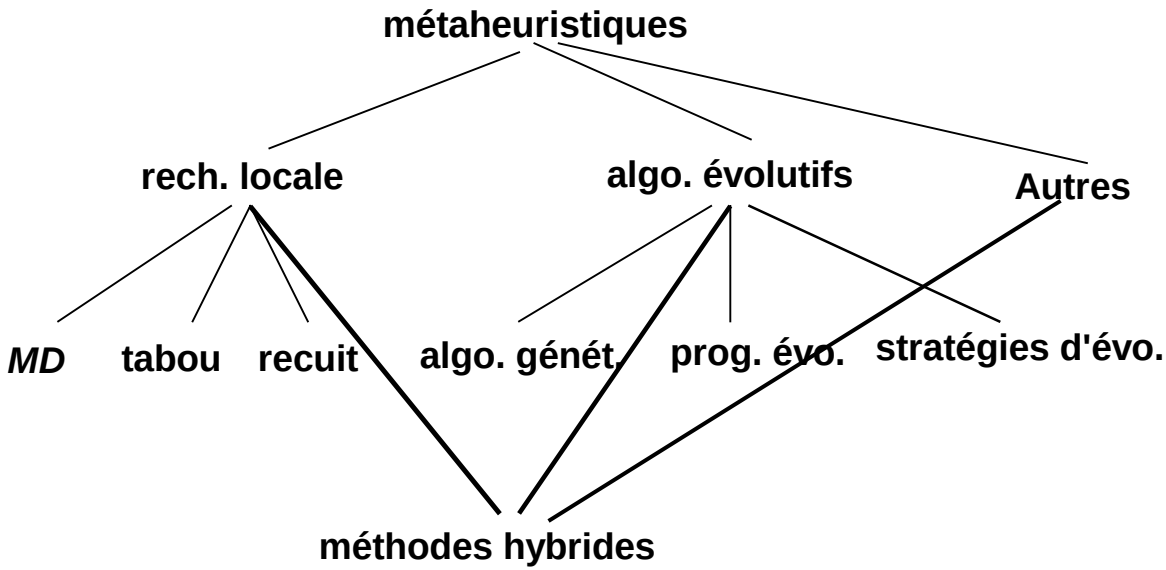
- réparation itérative d'une config. complète par des modifications locales
- non-exacte
- ex : descente, recuit simulé, recherche tabou, mais aussi min-conflit...

Remarque : possibilité d'insérer la recherche locale dans un algorithme exact (complet)

Approche d'évolution

- évolution d'une population de solutions par des opérations "génétiques" (sélection, croisement, et mutation)
- non-exacte
- ex : algo. génétiques, stratégies d'évolution, programmation évolutive...

Métaheuristiques



Remarques :

- domaine de recherche en pleine actualité et en évolution constante
- revue & conférence spécialisées
  * Metaheuristics International Conference (MIC) (depuis 1995)
  * Journal of Heuristics (depuis 1995, fondateur : F. Glover)

Heuristiques v.s. métaheuristiques

Heuristique

une méthode approchée conçue pour un *problème particulier* pour produire des solutions non nécessairement optimales (avec un temps de calcul raisonnable)

Métaheuristique

- une stratégie (règle) de choix pilotant une heuristique (introduite initialement dans le contexte de la méthode taboue)
- *un schéma de calcul heuristique, général et adaptable à un ensemble de problèmes différents*

Champs d'applications privilégiés

problèmes de recherche difficiles qu'on ne sait pas traiter autrement

Deux principes de base pour une recherche heuristique

Intensification (ou exploitation)

- permet d'examiner en profondeur une zone particulière de l'espace de recherche

Diversification (ou exploration)

- permet d'orienter la recherche vers de nouvelles zones (prometteuses) dans l'espace de recherche

Une recherche heuristique efficace

- nécessite un bon compris entre intensification et diversification

Remarque: Métaheuristiques fournissent des moyens différents pour la mise en oeuvre de ces deux principes complémentaires.

Recherche locale – éléments de base

Voisinage

- fonction $N : S \rightarrow 2^S$, $\forall s \in S$, $N(s) \subset S$,
  i.e. cette fonction associe à chaque point de S un sous-ensemble de S
  e.x. pour les CSP, changement de la valeur d'une variable
- $s \in S$ est un optimum local par rapport à N si $\forall s' \in N(s)$, $f(s) \leq f(s')$ (pour pb. min.)

Fonction d'évaluation Eval

- fonction *Eval* : $S \rightarrow R$, $\forall s \in S$,
  i.e. cette fonction permet de quantifier la qualité des points de S
- *Eval* = *f* (objectif initial) si l'on travaille dans l'espace de recherche réalisable
- *Eval* = *f* + une fonction de pénalité si l'on travaille dans l'espace non réalisable
- *Eval* = *f* + autres fonctions

Stratégie de mouvement

- règle de choix permettant de passer d'une configuration courante
  à une configuration voisine.

Remarque : La fonction d'évaluation (souvent négligée) est essentielle pour une recherche efficace.

Recherche locale – la procédure

Procédure générale

étape 1 (initialisation)

a) choisir une solution initiale $s \in S$

b) $s^* \leftarrow s$ (i.e. s^* mémorise la meilleure solution trouvée)

étape 2 (choix et terminaison)

a) *choisir* $s' \in N(s)$

b) $s \leftarrow s'$ (i.e. remplacer s par s')

c) terminer et retourner la meilleure solution trouvée si la condition d'arrêt vérifiée

étape 3 (mise à jour)

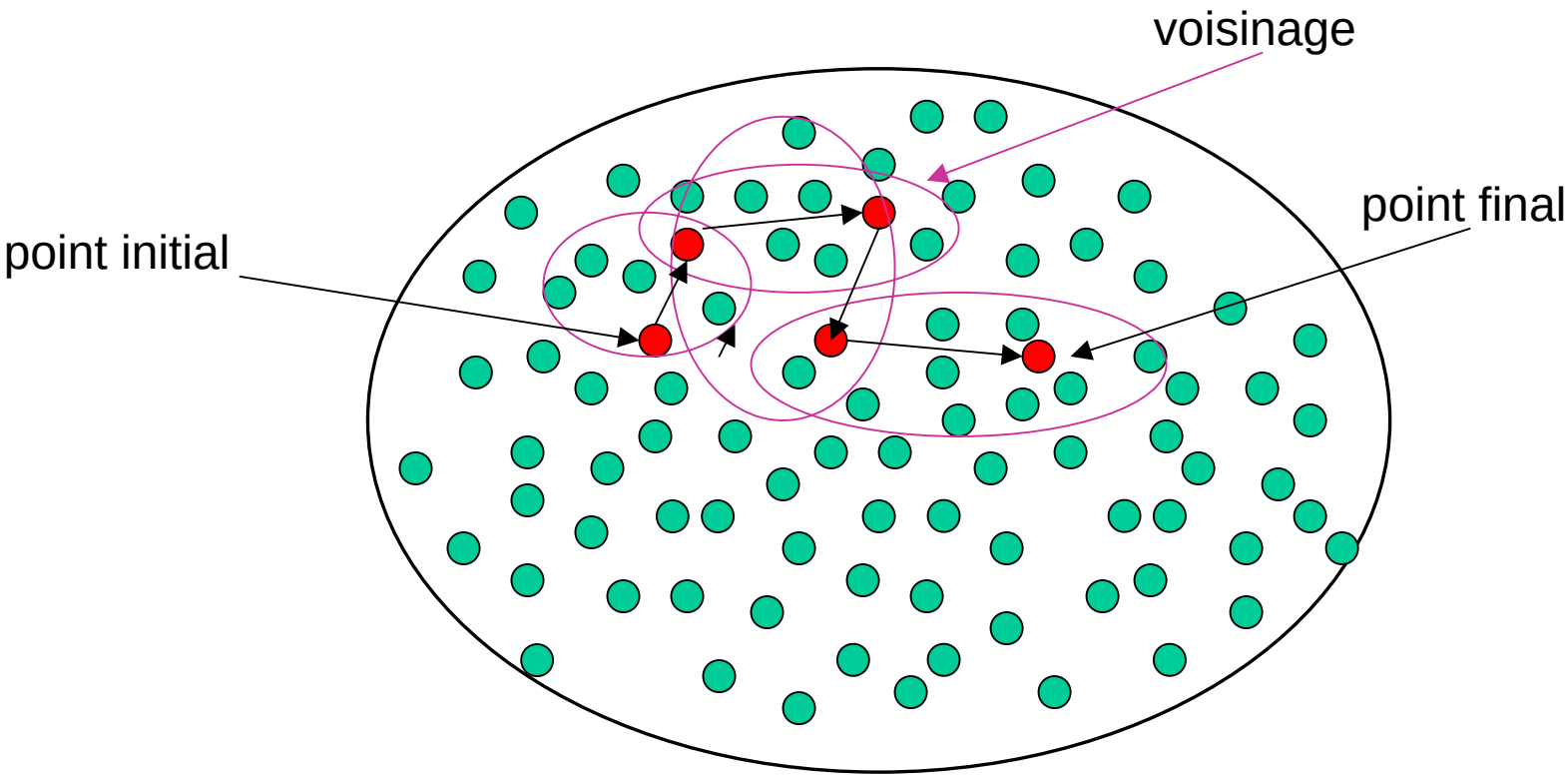
a) $s^* \leftarrow s$ si $f(s) < f(s^*)$

b) aller à l'étape 2

Remarque :

- comment choisir la solution initiale : heuristiques gloutonnes rapides ou aléatoire
- métaheuristiques différentes => stratégies de choix différentes à l'étape 2

Recherche locale



Recherche locale – la descente

Méthode de descente stricte

étape 2 (choix et terminaison)

Choisir un voisin s' $\in$N(s) tq f(s') < f(s) (s' est plus performant que s)
s $\leftarrow$ s' (i.e. remplacer s par s')
terminer si $\forall$ s'$\in$N(s), f(s') > f(s)

Remarque :

décisions à prendre

- *1ère ou meilleure amélioration
- comment évaluer rapidement les voisins à chaque itération (technique *delta, matrice de gains*)

Propriété : s'arrête au 1er optimum local rencontré

remèdes simples :

- relance multiple aléatoire, side-walk
- acceptation de voisins moins performants

Recherche locale – recuit simulé

Recuit simulé

étape 2 (choix et terminaison)

1. choisir *au hasard* s' $\in$ N(s) (s' n'est pas nécessairement meilleur que s)
2. si f(s') $\leq$ f(s), alors accepter s, sinon accepter s avec une *probabilité* p(Δf,T)
3. terminer si la condition d'arrêt est réalisée (un nb. max d'itér est effectué...)

Remarque :

- **décisions à prendre**
  1. comment déterminer la probabilité p(Δf,T) (diminution périodique ou continu)
  2. comment évaluer rapidement les voisins à chaque itération (*delta*)

- une recherche partiellement fondée sur le hasard

Schéma de refroidissement

Schéma de refroidissement (cooling schedule): la fonction qui spécifie l'évolution (souvent la diminution progressive) de la température

Schéma de décroissance par palier:

1. la température est conservée pour un nombre d'itération variables (paliers), la longueur de chaque palier est inversement proportionnelle à la valeur de la température

Autres schémas de refroidissement :

1. décroître la température à chaque itération.
2. une température constante (algorithme de Metropolis).
3. fonction de température non-monotone.

Le critère de Metropolis

Fonction CritMetropolis(Δf,T)

- **Si** Δf < 0 **renvoyer** VRAI

- **Sinon**

    avec une probabilité de exp(Δf,T) **renvoyer** VRAI

    Sinon renvoyer FAUX

Commentaires:

1. Un voisin qui améliore (Δf <0) ou à coût égal (Δf = 0) est toujours accepté.
2. Une dégradation faible est acceptée avec une probabilité plus grande qu'une dégradation plus importante.
3. La fonction CritMetropolis(Δf,T) est une fonction stochastique : appelée deux fois avec les mêmes arguments, elle peut renvoyer tantôt «vrai» et tantôt «faux».

Recherche locale – recuit simulé (avec paliers et refroidissement géométrique)

Algorithme général

Générer une configuration initiale s ;
s* ← s;
T := T0;
Repeat
    nb_moves := 0
    For i := 1 **to** iter_palier
        choisir *au hasard* s' ∈ N(s);
        calculer Δf = f(s') – f(s);
        if CritMetropolis(Δf, T) **then**
            s ← s';
            if f(s)<f(s*) **then** s* ← s;
    nb_moves := nb_moves + 1;
    acceptance_rate := i / nb_moves;
    T := T * coeff;
Until <condition fin>
Retourner s*

Recherche locale – tabou

Recherche tabou

étape 2 (choix et terminaison)

1. choisir s' $\in$ N(s) tq $\forall$ s'' $\in$ N(s), f(s') $\leq$ f(s'') (s' est le plus performant des voisins de s, mais peut être moins performant que s)
   (*liste tabou* pour empêcher des cycles)
2. s <- s', même si f(s') > f(s)
3. terminer si la condition d'arrêt est réalisée (nb. max d'itér effectué...)

Remarque :

- La règle de choix permet à tabu d'aller au delà des optima locaux
- **décisions à prendre**
  1. que stocker dans la liste tabou
  2. comment déterminer la longueur de la liste (statique ou dymanique)
  3. comment évaluer rapidement les voisins à chaque itération (*delta*)
- Mécanismes spécifiques pour intensification et diversification

Recherche locale (résumé)

Descente :

choisir un voisin plus performant $s' \in N(s)$ i.e. $f(s') < f(s)$
Rapide mais s'arrête au 1er optimum local rencontré

Recuit :

choisir au hasard $s' \in N(s)$; si $f(s') \leq f(s)$ alors accepter s'
sinon accepter s' avec probabilité $p(\Delta f, T)$

Tabou :

choisir un des meilleurs voisins $s' \in N(s)$, accepter s' même si $f(s') > f(s)$
(*liste tabou* pour empêcher des cycles)

- Recuit et tabou ne s'arrêtent pas au 1er optimum local rencontré

D’autres formes de recherche locale

Recherche à Voisinage Variable (VNS – Variable Neighborhood Search)

La descente s’opère avec un *ensemble* de voisinages N_k (ex: dans le cas d’un problème 0-1, $N_k(x)$ = { distance_hamming(x,x’)=k}

GRSAP (Greedy randomized adaptative search procedures)

La descente (ou une autre RL) est appliquée sur des solutions initiales générées avec une procédure *gloutonne non-déterministe*.

Randow-Walk

Avec une probabilité p, prendre un voisin au hasard
Avec une probabilité 1-p, appliquer la descente
Un exemple : le fameux algorithme *WalkSAT* (GSAT est une descente stricte)

Méthode bruitage (Noising method)

La recherche locale est appliquée avec une fonction d’évaluation *f* bruitée ou sur les données bruitées du problème

Recherche locale : performance

Théorique

preuves de convergence dans certains cas (recuit simulé, tabou...), mais difficlement utilisables en pratique

Pratique

- très bons résultats pour de nombreux problèmes difficiles
- *adaptation* indispensable :
  - le codage du problème (configuration et espace de recherche)
  - le voisinage (connaissances spécifiques du problème)
  - techniques pour traiter les contraintes
  - les structures de données employées

Amélioration

- hybridation avec les algorithmes génétiques
- hybridation avec l'approche de construction

Approche d'évolution (à population)

Notion principale

- évolution d'un ensemble de configurations (*population*)
- opérateurs d'évolution : sélection, recombinaison et mutation

Procédure générale

étape 1 : (*initialisation*)

Choisir un ensemble de configurations initiales (population)

étape 2 : (*évolution*)

Sélection sur la population

Application d'opérateurs de recombinaison et de mutation

étape 3 : (*mise à jour*)

Réorganisation de la population (ex, élimination des configurations non-performantes de la population)

Remarques :

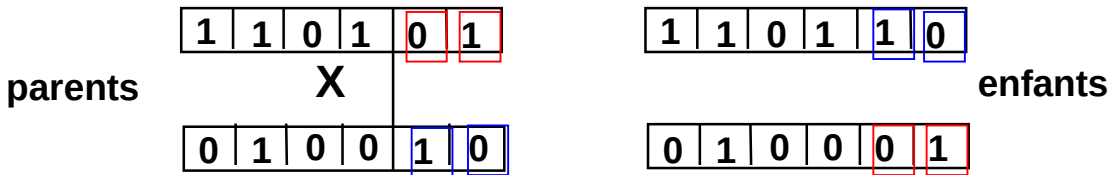
- différentes écoles : AG, stratégies d'évolution et program. évolutive
- un cadre algorithmique très général et puissant

Algorithme génétique simple (Holland 75)

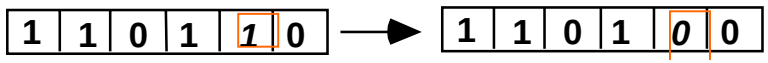
Caractéristiques :

- codage universel du problème sous forme de chaînes en 0/1
- opérateurs "génétiques" stochastiques (aveugles) : mutation et croisement

Croisement : échange de bits entre deux config. (monopoint, bi-points, uniforme...)



Mutation : modification de valeurs de bits déterminés au hasard d'une config. (enfant)



Remarques :

justification par le théorème des schémas :

- les schémas courts et performants (building blocks) se X et se multiplient
- la performance d'une config. est un indicateur directe de la performance de tous les schémas auxquels elle appartient (// implicite)

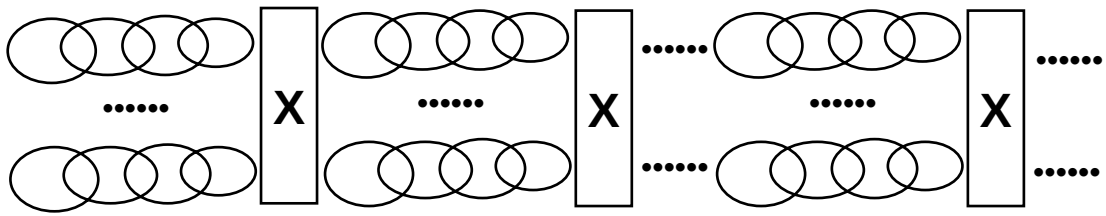
décisions à prendre :

- comment évaluer *rapidement* les configurations de la population
- comment équilibrer l'exploitation et l'exploration

Approche d'évolution

La pratique

- **Spécialisation** :
  - configuration (codage) *adapté* au problème (ex, permutation pour TSP)
  - opérateurs d'évolution (mutation et croisement) *adaptés* au problème
- **Hybridation** :
  - avec l'approche de construction
  - avec la recherche locale



Remarque: intégration indispensable de connaissances spécifiques

Approche d'évolution : performance

Théorique

- preuves de convergence dans certains cas, difficilement utilisables en pratique

Pratique

- résultats souvent faibles avec *AG naïfs*, i.e. mutation et croisement *aveugles*

- résultats fortement compétitifs avec *AG spécialisés*
  - codage adapté au problème
  - opérateurs "génétiques" intégrant des connaissances du problème

- résultats fortement compétitifs, même les meilleurs avec *AG hybrides*
  - hybridation avec l'approche de construction (ex, glouton)
  - hybridation avec la recherche locale : *GLS - Genetic Local Search*

Remarque :

- l'adaptation est indispensable

Approche hybride – génétique recherche locale

Idée de base :

combiner 2 méthodes complémentaires : recherche *globale* et recherche *locale*
- diversification avec recombinaison (spécifique)
- intensification avec recherche locale

2 Schéma d'hybridation

- AG + descente :
  les configurations de la population sont des optima locaux.

- AG + tabou (recuit) :
  les configurations de la population sont améliorées pendent un nombre fixe d'itérations.

Principe de conception

- l'opérateur de RL doit être performant
- l'opérateur de croisement doit être spécialisé au problème

Algorithme génétique hybride

Procédure générale

étape 1 (initialisation)

a) générer une population de configurations P

b) **appliquer une recherche locale à chacune des configurations de P**

étape 2 (évolution et terminaison)

a) choisir **p1** et **p2** dans P

b) générer une configuration e par une recombinaison de **p1** et **p2**

c) **améliorer e par l'application de la recherche locale**

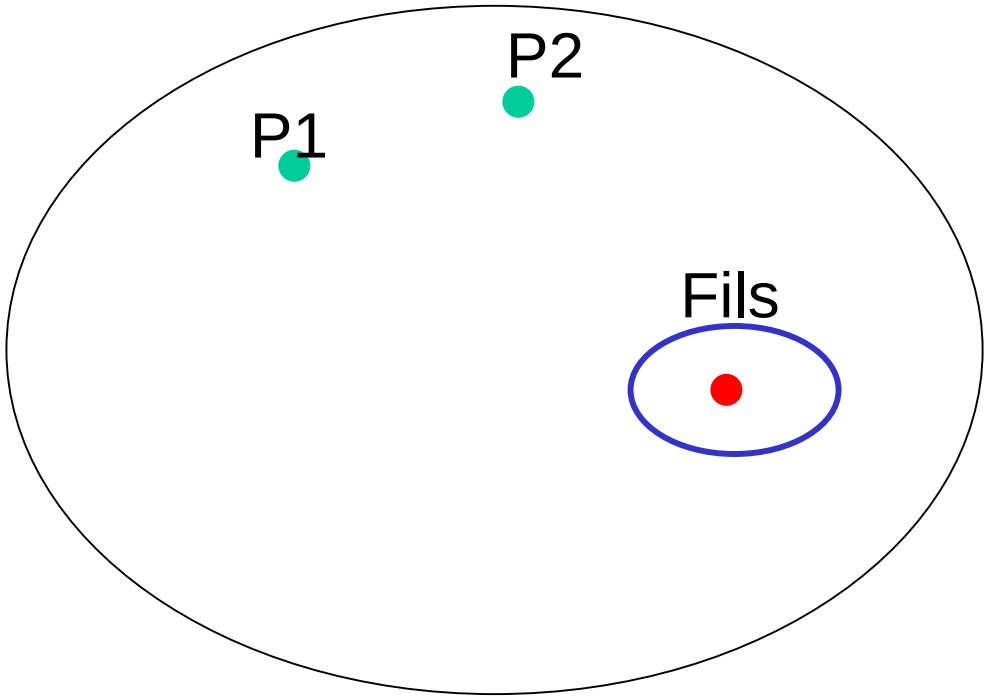
d) Insérer la nouvelle configuration **e** dans la population

e) terminer et retourner la meilleure solution trouvée si la cond. d'arrêt vérifiée

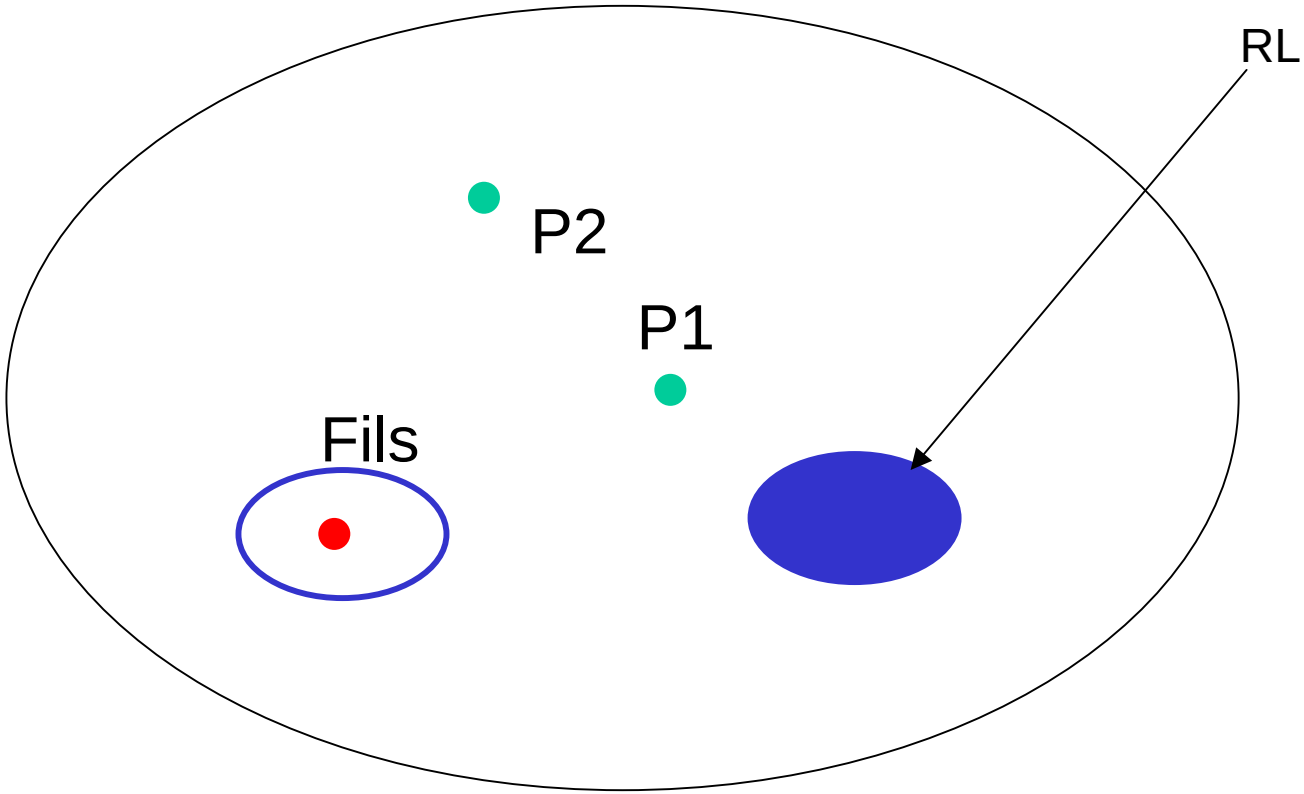
étape 3 (*mise à jour*)

a) re-organisation de la population (ex, élimination des configurations non-performantes de la population)

Génétique recherche locale



Algorithme génétique hybride



D’autres approches fondées sur la notion de population/agent

La recherche dispersée (Scatter Search)

- Recombinaison de plusieurs solutions
- Remplacement de solutions fondé sur la qualité et la diversité

Optimisation par colonie de fourmis

Optimisation par essaim particulaire (Particle Swarm Optimization)

Adaptation des méta-heuristiques

Résolution d'un problème avec une métaheuristique

- analyse et modélisation du problème à traiter
- choix d'une métaheuristique selon
  - la qualité des solutions recherchées
  - la disponibilité des connaissances sur le problème
  - le savoir-faire...
- adaptation de la métaheuristique au problème
  - configuration (espace de recherche)
  - fonction d'évaluation
  - voisinage
  - opérateurs de recherche
  - traitement des contraintes
  - structures de données...

Évaluation de l'algorithme (*benchmarking* si possible)

- qualité de la meilleure solution trouvée ou profile de recherche
- rapidité, *i.e.* l'effort (le "temps" de calcul, le nb d'itération...) nécessaire pour trouver une solution
- robustesse

Conclusions sur les métaheuristiques

Atouts

- générales et applicables à une très large classe de problèmes
- possibilité de compromis entre le temps de calcul et la qualité de solution
- possibilité d'intégrer des connaissances spécifiques du problème
- domaines d'application privilégiés : problèmes combinatoires de grande taille

"Inconvénients"

- optimum global non garanti
- adaptation souvent indispensable
- difficulté de prévoir la performance (qualité et temps)

Performance

- **théorique**: preuve de convergence dans certains cas, non utilisable en pratique
- **pratique** : dépend de l'adaptation au problème (codage du problème, connaissances spécifiques, traitement de contraintes, structures de données...)

Perspectives:

- Systèmes de résolution *génériques* fondés sur les métaheuristiques

- ***L'approche méta-heuristique constitue un outil puissant pour la résolution de problèmes combinatoires difficiles (discrets et continus)***

Deuxième partie

Exemples d'application

Les applications citées ont été réalisées avec mes collaborateurs

Exemples d'application

Applications

- positionnement d'antennes pour les réseaux radio-mobiles (Esprit 4)
- affectation de fréquences dans les réseaux radio-mobiles (France Telecom)
- planification d'interventions de techniciens (France Telecom)
- planification journalière de prises de vues d'un satellite (Application CNES)
- évolution d'équipements de réseaux télécoms (Bouygues Telecom)
- planification intégrée de chauffeurs et de véhicules
- planification de rencontres sportives

Problèmes de référence

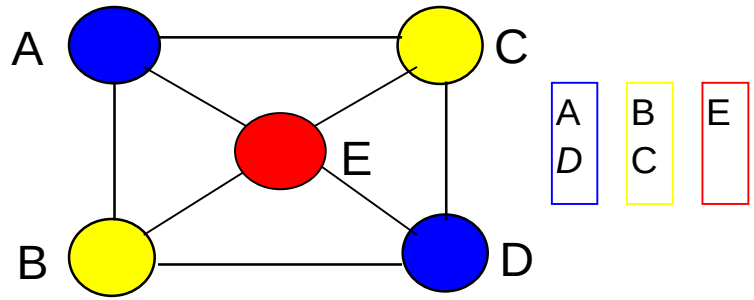
- coloration et T-coloration de graphes
- minimisation de largeurs de bande
- minimisation d'arrangements linéaires
- sac-à-dos multidimensionnel
- planification de rencontres d'équipage (*Progressive Party Problem*)
- satisfaction de contraintes CSP et MCSP

- reconstruction d'arbre phylogénétique

Coloration de graphes

k-coloration (satisfaction)

étant donné un graphe G=<V,E>, trouver une partition de V en k classes telle que 2 nœuds adj. ne soient pas dans la même classe (k-coloriable)



Coloration (optimisation)

- déterminer le plus petit k pour lequel le graphe est k-coloriable (le nb chromatique)

Intérêts du problème

- beaucoup d'applications : affectation de fréquences, emploi du temps, ordonnan…
- beaucoup de travaux existants
- grande difficulté : NP-difficile (impossible de colorier de manière exacte des graphes aléatoires de densité 0.5 > 90 nœuds, d'après Johnson et al. 91)
- un des 3 problèmes traités par le "2nd Dimacs Implementation Challenge" (92-93)

Coloration de graphes

Algorithmes connus

construction

DSATUR (Brélaz 79)
RLF (Recursive largest First) (Leighton 79)
XRLF (Johnson et al. 91)

recherche locale

recuit simulé (Chams et al. 87, Johnson et al. 91)
tabou (Hertz & de Werra 87, Dorne & Hao 97)

Hybridation

population + RL + construction (Morgenstern 94)
croisement d'affectation et RL (Fleurent & Ferland 94, Costa et al. 95)
croisement spécialisé et RL (Dorne & Hao 98, Galinier & Hao 99)

Algorithme génétique locale pour la coloration

Processus de résolution générale

1. On fixe le nombre de couleurs k, on cherche une k-coloration propre
2. Si succès, on décrémente k, on va à l'étape 1)
3. On s'arrête quand on n'arrive plus à trouver une k-coloration

Définitions

- espace de recherche S : toutes les partitions de V en k classes

$$S = \{s \mid s \text{ est une partition de V en k classes } \{V_1, V_2, ..., V_k\}\}$$

- fonction d'évaluation f : le nombre d'arêtes ayant deux extrêmes coloriées avec une même couleur

$$\forall s \in S, f(s) = |\{<u,v> \in E \mid u \in Vi \text{ et } v \in Vi\}$$

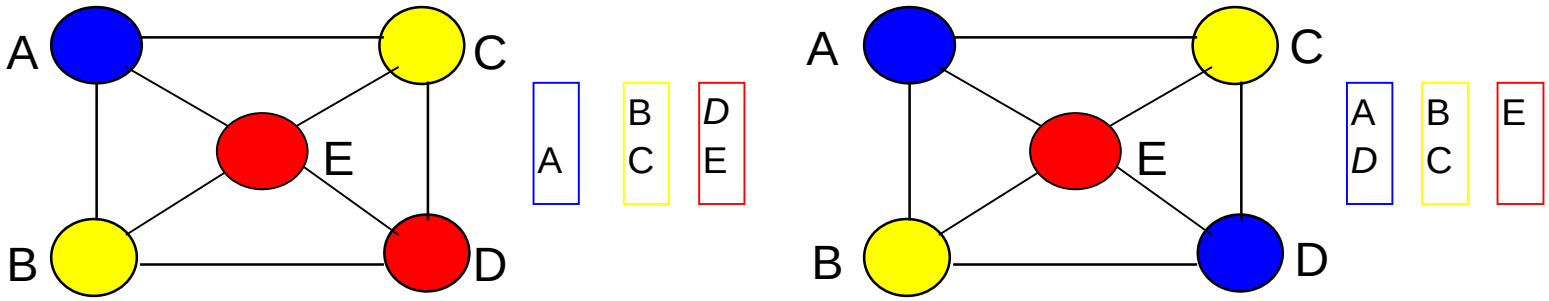
Remarque :

f(s) = 0 si s est une coloration propre (sans conflit) pour une valeur de k fixe

Algorithme génétique locale pour la coloration

Recherche locale avec tabou

- *voisinage* : on change la classe d'un nœud en conflit



- *liste tabou* :
  - *contenu* : quand un nœud u perd sa couleur c(u), u est interdit de reprendre c(u) pendant l(u) itérations
  - *longueur* : l(u) définie dynamiquement en fonction du nombre de nœuds en conflit.

- *évaluation des configurations* :
  - structure de données performantes
  - algorithmes incrémentaux pour une évaluation rapide des voisins.

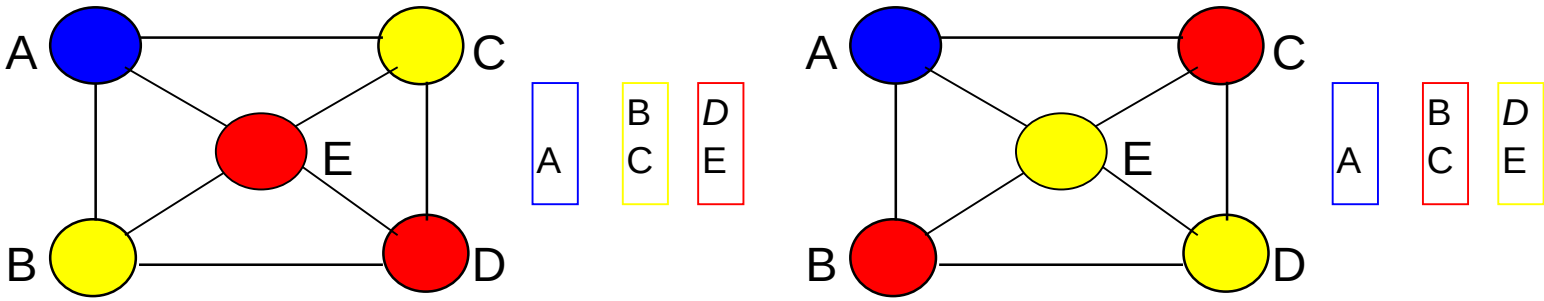
Algorithme génétique locale pour la coloration

Croisements classiques

uniforme : la couleur d'un nœud chez un enfant est déterminée aléatoirement par la couleur de l'un des parents

affectation : la couleur d'un nœud est déterminée d'après les règles suivantes:
si les deux parents ont la même couleur, alors prendre cette couleur
si les deux parents n'ont pas la même couleur, alors prendre aléatoirement la couleur de l'un des parents (ou une autre couleur)

Remarque :
Problème de permutation de couleurs (symétrie); ces croisements ne permettent pas de transmettre les *bonnes propriétés* des parents aux enfants.



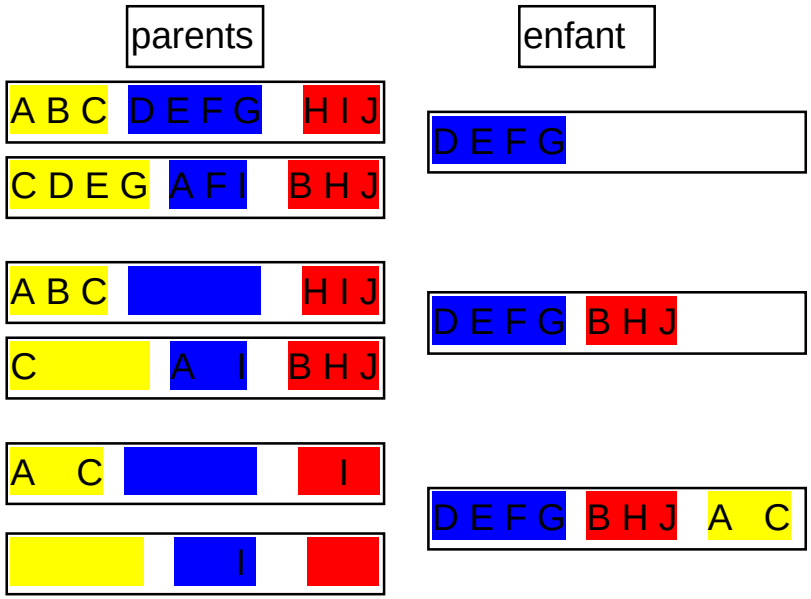
Algorithme génétique locale pour la coloration

Croisement spécialisé GPX (*Greedy Partition Crossover*)

(P. Galinier et J.K. Hao, Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4): 379-397, 1999.)

- Une bonne propriété (sémantique) est la suivante :
  - Les classes de grand cardinal doivent être conservées chez l'enfant.

- Le croisement GPX
  - pour chaque classe l pair, transmettre la classe de cardinal maximum de p1
  - pour chaque classe l impair, transmettre la classe de cardinal maxi. de p2

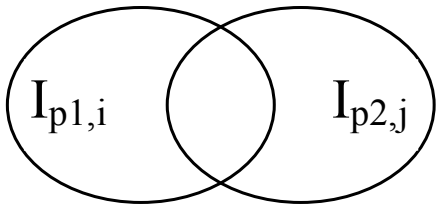


Algorithme génétique locale pour la coloration

Croisement spécialisé UIS (*Union of Independent Sets*) (R. Dorne et J.K. Hao, A new genetic local search algorithm for graph coloring. *LNCS* 1498 : 745-754, Sept. 1998)

- Une autre bonne propriété est la suivante :
  - si deux classes sans conflit $V^1_i \in p1$ et $V^2_j \in p2$ partagent un nombre important de nœuds, on doit conserver les deux classes chez l'enfant.

- Le croisement UIS
  - $I_{p1,i} \subseteq V^1_i \in p1$ est un ensemble indépendant de cardinal maximum chez p1, on cherche chez p2 un ens. indépendant $I_{p2,j} \subseteq V^2_j \in p2$ tel que $\forall\ j \in \{1..K\}$, $|I_{p1,i} \cap I_{p2,j}|$ soit maximal
  - on construit une classe i chez l'enfant par l'union de $I_{p1,i}$ et $I_{p2,j}$
  - les nœuds absents sont distribués aléatoirement



Remarque : un ou deux enfants peuvent être générés

Résultats sur des benchmarks

Jeux de test (DIMACS benchmarks)

- graphes construits (Leighton 79, Culberson 79)
- graphes aléatoires $G_{n,d}$ (Hertz & de Werra 87, Johnson *et al.* 91)

Critères d'évaluation

- qualité de la meilleure solution (la plus petite valeur pour k)
- rapidité en terme de nombre d'itérations et en temps

Paramètres

- taille de la population : 5 ou 10
- solutions initiales : construites avec l'algorithme de DSATUR
- nombre d'itérations pour chaque RL : 250 à 4000
- liste tabou : réglage dynamique et automatique
- condition d'arrêt : une k-coloration trouvée ou nb max. d'itérations atteint

Résultats sur des benchmarks

Table 6. Comparison between HCA and the best known results.

Graph	χ	Best-known	TS	HCA	Difference
DSJC250.5	—	28	28	28	0
DSJC500.5	—	48	49	48	0
DSJC1000.5	—	84	89	83	−1
le450_15c	15	15	16	15	0
le450_25c	25	25	26	26	1
flat300_28	28	31	32	31	0
flat1000_76_0.col	76	84	87	83	−1
DSJC1000.1	—	21	—	20	−1
DSJC1000.9	—	226	—	224	−2

Extrait du papier : P. Galinier et J.K. Hao, Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization, 3(4): 379-397, 1999.

Problèmes de T-coloration

T-coloration simple

- étant donné un graphe G=<V,E> et un ensemble Tij d'entiers positifs incluant la valeur 0 pour chaque (Vi,Vj)$\in$E, trouver une fonction c : V-> {1,2...k} telle que :

$$\forall\ (Vi,Vj)\in E,\ |\ c(Vi) - c(Vj)\ | \notin Tij$$

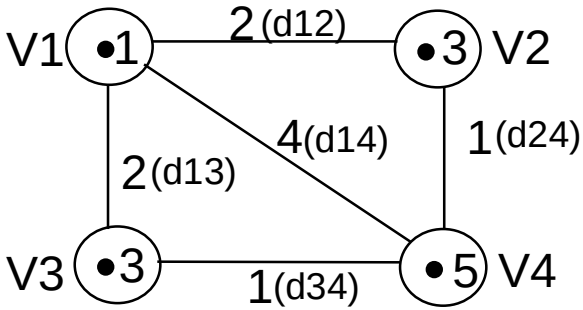
(la distance entre chaque paire de couleurs de 2 noeuds adj. $\neq$ des valeurs de Tij)

Remarques :

- si Tij = {0,1...dij}, alors $\forall$ (Vi,Vj)ÎE, $|\ c(Vi) - c(Vj)\ | \geq$ dij

- "T-Span" d'une T-coloration : la distance entre la couleur max. et la couleur min.
- "T-Span" d'un graphe G : le plus petit "T-span" de tts les T-colorations poss. de G

T-coloration



T-span = 4

Problèmes de coloration

T-coloration avec ensembles

À chaque noeud Vi, on associe :

- un entier bi $\geq$ 1, le nb de couleurs demandé par Vi et
- un ensemble Ti d'entiers incluant la valeur 0 (val. interdites entre paire de couleurs)

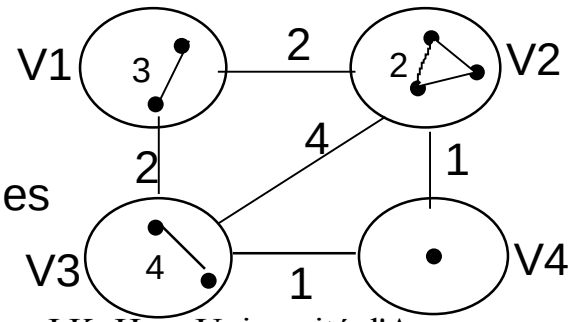
Colorier chaque noeud Vi $\in$V avec bi $\geq$1 couleurs tout en respectant les contraintes :

- la distance entre chaque paire de couleurs d'un même noeud Vi doit être différente des valeurs de Ti : $\forall$ Vi $\in$ V, m, n $\in${1...bi}, m $\neq$ n, | c(Vim) - c(Vin) | $\notin$ Ti,
- la distance entre chaque paire de couleurs de 2 noeuds adj. Vi et Vj doit être différente des valeurs de Tij : $\forall$ (Vi,Vj) $\in$ E, mÎ{1...bi}, n $\in${1...bj}, |c(Vim) - c(Vjn)| $\notin$ Tij

Remarques :

- si Ti = {0,1...di}, alors $\forall$Vi $\in$ V, m,nÎ{1...bi}, m $\neq$ n, |c(Vim) - c(Vin) $\geq$ di
- affectation de fréquences est une application concrète de la T-coloration

T-coloration avec ensembles



Problèmes de coloration

Méthodes de résolution

- coloration
  - construction : DSATUR (Brélaz 79), RLF (Recurs. largest First) (Leighton 79)
  - recuit simulé (Chams et al. 87, Johnson et al. 91), tabou (Hertz & de Werra 87)
  - hybridation (Fleurent & Ferland 94, Morgenstern 94, Costa et al. 95)
- T-coloration simple
  - ordre dynamique (glouton) (Gamst 92)
  - tabou et recuit simulé (Costa 93)
- T-coloration avec ensembles
  - ordre dynamique (glouton) et tabou (Jiang 96)

Algorithmes récents

- Algorithme hybride “génétique + tabou” pour la coloration
- Algorithme tabou pour T-coloration et T-coloration avec ensemble

Problèmes de coloration

Jeux de test

- coloration
  - DIMACS benchmarks
  - graphes de taille jusqu'à 1000 noeuds et 200 couleurs environ

- T-coloration
  - jeux aléatoires (pas de benchmark disponible)
    - + distance de séparation prise entre 1 et 5
    - + nb de couleurs par noeud pris entre 1 et 5
  - graphes de taille jusqu'à 1 000 noeuds (environ 3 000 variables entières, 4 millions de contraintes) et 2 000 couleurs

Remarque **:** d'après (Johnson et al. 91), aucun algo. exact exist. ne peut colorier des graphes aléatoires de densité 0.5 > 90 noeuds

Problèmes de coloration

Résultats

- coloration
  - les algorithmes tabou et hybrides dominent les algo. de construction
  - AG hybrides améliorent les meilleurs résultats de certains jeux DIMACS

- T-coloration
  - les algorithmes tabou et hybrides dominent l'algorithme DSATUR adapté
  - les résultats de l'algo. tabou sont proches de l'estimation théorique

Remarques :

- intéressant d'initialiser la recherche avec un algo. de construction

SAT & Max-SAT

Définition SAT

- étant donnée une conjonction de clauses, déterminer une affectation des valeurs {0,1} à toutes les variables qui rend toutes les clauses vraies.

- si toutes les clauses ne peuvent pas être satisfaites, alors déterminer une affectation qui maximise le nombre de clauses vraies (Max-SAT).

Remarque : un des problèmes NP-Complet les plus étudié avec des applications variées (VLSI, emploi du temps, …).

SAT & Max-SAT

Méthodes de résolution

- algorithmes complets
  - * Davis & Putnam (depuis 1960) : SATZ (1997), Zchaff (2002)...

- algorithmes incomplets
  - * recherche locale : tabou, recuit simulé
  - * GSAT & WalkSAT (92, 94), unitWalk (2002)
  - * hybride "génétique et tabou" GASAT (2003)

GASAT

- un **croisement spécifique** fondée sur la satisfaction de clauses
  - * corriger les clauses fausses
  - * conserver les clauses vraies
- un algorithme tabou

Problème de rencontres d'équipages
(*PPP - Progressive Party Problem*)

Le problème

affecter 29 équipages visiteurs de tailles différentes à 13 bateaux hôtes de capacités diff. pour T périodes de temps consécutives avec les contraintes :

- chaque équipage visiteur visite un bateau différent pour chaque période différente.
- deux équipages ne doivent pas se rencontrer plus d'une fois
- la capacité de chaque hôte doit être respectée

Remarques :

- problème de base avec T = 6, plus difficile quand T augmente,
- pas de solution connue pour T > 8 (en 1996), mais T= 10 une borne sup.
- contraintes hétérogènes, non binaires

Problème de rencontres d'équipages
(*PPP - Progressive Party Problem*)

Méthodes de résolution

- programmation linéaire en nombres entiers : échec pour T<6 (Brailsford et al. 96)
- program. par contraintes 1: T = 6 et 7 en 20-30 mn, échec pour T ≥ 8 (Smith et la 96)
- program. par contraintes 2 : T < 7 en quelques sec., T = 8 en quelques heures, échec pour T>9 (Cosytec 97)

Approche heuristique avec tabou (Galinier & Hao 98)

- une formalisation du problème en CSP :
  - * var. = un couple (équipage, période) et dom. = l'ens. des hôtes D = {1...13}
  - * contraintes : n-aires
- deux voisinages :
  - * N1 : changer la valeur d'une seule variable en conflit (équipage, période) (changer l'hôte d'un équipage qui viole des contraintes)
  - * N2 : échanger les valeurs de 2 variables (une au moins est en conflit) (échanger les hôtes de 2 équipages d'une même période)
- fonction d'évaluation : basée sur une agrégation de pénalité
- algorithmes de recherche locale (descente, metropolis, tabou)

Problème de rencontres d'équipages
(*PPP - Progressive Party Problem*)

Résultats

- pour T≤9, solutions en moins de 5 secondes
- échec pour T = 10 (une contrainte non satisfaite)

Remarques :

- résolution du pb jusqu'à T = 9 avec N1 et N2
- N2 plus performant car il permet de résoudre le pb. jusqu'à T= 8 avec la descente
- pas de différence significative entre métaheuristiques utilisées (recuit et tabou)

- problème ouvert pour T = 10

=> *jusqu'à T = 9, problème simple pour la recherche locale*

Planification de tournois sportifs
(*SLSP – Sports League Scheduling Problem*)

Le problème

Planifier les tournois entre *T* équipes (*T* pair) sur *T-1* semaines, chaque semaine étant découpée en *T/2* périodes avec les contraintes suivantes :

- *unicité / semaine* : chaque équipe joue exactement 1 fois / semaine,
- *double / période* : aucune équipe ne peut jouer plus de 2 fois / période,
- *tournoi simple* : chaque équipe joue contre toutes les autres exactement une fois.

	Semaine 1	**Semaine 2**	**Semaine 3**	**Semaine 4**	**Semaine 5**	**Semaine 6**	**Semaine 7**
Période 1	**1 vs 2**	**1 vs 3**	**5 vs 8**	**4 vs 7**	**4 vs 8**	**2 vs 6**	**3 vs 5**
Période 2	**3 vs 4**	**2 vs 8**	**1 vs 4**	**6 vs 8**	**2 vs 5**	**1 vs 7**	**6 vs 7**
Période 3	**5 vs 6**	**4 vs 6**	**2 vs 7**	**1 vs 5**	**3 vs 7**	**3 vs 8**	**1 vs 8**
Période 4	**7 vs 8**	**5 vs 7**	**3 vs 6**	**2 vs 3**	**1 vs 6**	**4 vs 5**	**2 vs 4**

Exemple d'une programmation valide pour 8 équipes

Remarque: problème se résout en temps linéaire si (*T* -1) MOD 3=0 (T=10, 16, 22, 28, 34, 40, 46)
problème encore ouvert dans le cas général (T=12, 14, 18, 20, 24, 26, 30, 32, 36, 38, 42, 44...)

Planification de tournois sportifs
(*SLSP – Sports League Scheduling Problem*)

Méthodes de résolution

- PLEN avec contraintes de cardinalité (Cplex) : $T \leq 12$ (McAloon et al. 97)
- PPC avec contraintes de différence (ILOG Solver) : $T \leq 14$ (McAloon et al. 97)
- PPC avec algorithmes de filtrage puissants (ILOG Solver) : $T \leq 24$ (Régin 98)
- PPC avec transformation du problème (ILOG Solver) : $T \leq 40$ (Régin 99)

Approche heuristique avec tabou (Hamiez & Hao 00)

- une formalisation du problème en CSP
- une heuristique pour la construction d'une solution initiale
- voisinage fondé sur échange entre deux matches (dont un est en conflit)
- traitement de contraintes : satisfaction constante et pénalité
- longueur de la liste tabou dynamique

Planification de tournois sportifs
(*SLSP – Sports League Scheduling Problem*)

Résultats

- solutions trouvées jusqu'à $T = 40$
- pour $T \leq 20$, solutions en quelques secondes

Remarques sur l'algorithme tabou:

- solution initiale joue un rôle important
- la façon de traiter les contraintes joue un rôle important
- diversification indispensable

Remarque sur le problème :

- beaucoup à faire pour résoudre le problème pour T> 40 équipes si (T-1) MOD 3 $\neq$ 0

Sac-à-dos multidimensionnel

Définition du problème

Etant donné n variables bivalentes et m contraintes linéaires

$$\max \vec{c} \times \vec{x}, \;\; s.c. \; A.\vec{x} \leq \vec{b},$$

avec

$$\vec{c} \in \mathrm{N}^{*n}, A \in \mathrm{N}^{m \times n}, \vec{b} \in \mathrm{N}^{*m} \; et \; \vec{x} \in \{0,1\}^n$$

Remarque **:**

- très nombreuses applications dans des domaines variés,
- NP-difficile, résolutions exactes limitées à des instances n < 90 et m < 5.

Sac-à-dos multidimensionnel

Méthodes de résolution

- algorithmes exacts (B&B) (Shih 79)
- relaxation (Fréville & Plateau 93)
- Tabou (Glover & Kochenberger 97, Hanafi & Fréville 98) (très bon résultats)
- génétiques spécialisés (Chu & Beasley 98)
- algorithme hybride “simplex + tabou” (Vasquez & Hao 00d)

Jeux de test

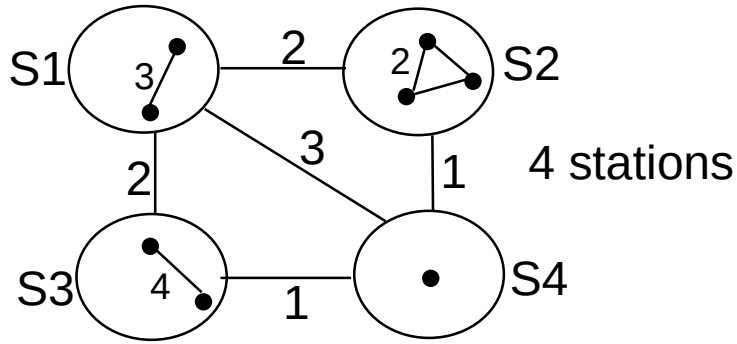
- benchmark OR-LIB et d’autres (n=100 à 2500 et m = 5 à 100)

Résultats

- L’algorithme hybride fournit les meilleurs résultats pour l’ensemble des benchmarks les plus difficiles.

Affectation de fréquences dans les réseaux radio-mobiles

Le problème



étant donné

1. n stations {S1, S2...Sn}
2. les trafics bi (i $\in$ {1...n}), i.e. le nombre de fréquences demandé par station
3. les contraintes d'interférences définies par une matrice de réutilisation M[n,n]
   - "co-station": $| f_{i,g} - f_{i,h}| \geq M[i,i], \forall\ i \in \{1...n\}, \forall\ g,h \in \{1..Ti\}, g \neq h$
   - "stations adjacentes": $| f_{i,g} - f_{j,h}| \geq M[i,j], \forall i,j \in \{1...n\}, i \neq j, \forall g \in \{1..bi\}, \forall h \in \{1..bj\}$

trouver un plan d'affectation

en minimisant les interférences avec k fréquences données (interférences mesurées en termes de contraintes violées)

Remarque : c'est un problème de coloration généralisé

Affectation de fréquences dans les réseaux radio-mobiles

Méthodes de résolution existantes

- recuit simulé (Duque-Anton et al. 93, CNET 95)
- réseaux de neurones (Kunz 91)
- algorithmes génétiques (Crompton et al. 94)
- algorithmes de coloration (Gamst 86, CNET 95)

Algorithmes Tabou et génétiques récents

(Dorne & Hao 95, Hao & Dorne 96, Renaud & Camanida 97, Hao et al. 98)

- Traitements de contraintes
- Croisement spécialisés

Affectation de fréquences dans les réseaux radio-mobiles

Un algorithme tabou

- *configuration* : une affectation respectant les trafics et *les contraintes de co-station*

S1		S2			S3		S4
f11	f12	f21	f22	f23	f31	f32	f41

- *fonction de voisinage N : S* $\rightarrow 2^S$ **:** s et s' voisins s'ils diffèrent par la valeur d'une seule fréqu. *en conflit* d'une station
- *liste tabou* : attribut mémorisé = <station, ancienne_val_de_fréqu.>
- *la longueur de la liste tabou* : une fonction linéaire de la taille du voisinage
- *évaluation incrémentale des configurations* : valeur d'évaluation de chaque mouvement possible (une matrice Δ de w x k)
- *aspiration* : accepter tout mouvement conduisant à une configuration de qualité supérieure à la meilleure configuration rencontrée

Affectation de fréquences dans les réseaux radio-mobiles

Un algorithme tabou

étape 1 (initialisation)

- choisir une solution initiale $s \in S$
- mémoriser la meilleure solution trouvée $s^* \leftarrow s$
- initialiser les structures de données (liste tabou, matrice Δ...)

étape 2 (choix et terminaison)

- choisir un des meilleurs voisins non tabou $s' \in N(s)$ tq $\forall\ s'' \in N(s)$, $f(s') < f(s'')$
- $s \leftarrow s'$ (même si s' est moins performant que s)
- terminer si max_itér est effectué (ou si s n'est plus améliorée pendant max_itér)

étape 3 (mise à jour)

- $s^* \leftarrow s$ si $f(s) < f(s^*)$
- rendre le dernier mouvement tabou pendant k itérations
- mettre à jour d'autres structures de données (matrice Δ...)
- aller à l'étape 2

Affectation de fréquences dans les réseaux radio-mobiles

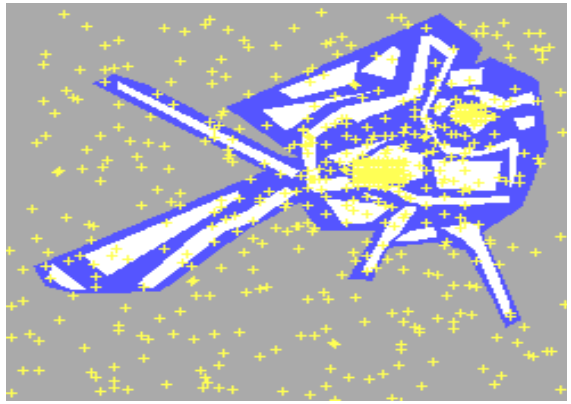
Jeux de test fictifs et réels (fournis par le CNET)

- nb de fréquences par station : 1 à 4
- distance de séparation de fréquences de co-station : 2 à 4
- distance de séparation de fréquences de stations adjacentes : 1 à 3
- de grande taille, jusqu'à
  - 1 000 variables entières
  - 54 valeurs par variables
  - 35 000 contraintes

Résultats

- L’algorithme tabou et l’algo. génétique spécialisé dominent largement l'approche classique de construction
- Les croisements standards jouent un rôle marginal

Positionnement d'antennes pour les réseaux radio-mobiles



Réseaux urbain : 50 km x 46 km
568 sites potentiels
56792 points de mesure du signal radio
17393 STP
6652 TTP pour 2988.08 'erlang'

Zone géographique

- un ensemble S de points de service (STP en **bleu**) : seuil de qualité **Sq** du signal radio défini par un niveau de champ (-82 dBm→2W'incar',-90 dBm→8W 'outdoor',...),
- un ensemble $T \subset S$ de points de trafic (TTP en blanc) : estimation de trafic en erlang,
- un ensemble L de sites candidats pour le positionnement d'antennes

Matrice de perte en propagation du signal radio

- pour chaque site les valeurs de l'atténuation du signal radio sur tous les STP

Antennes

- différents types : omnidirectionnelle, directionnelle large ou petite ouverture
- paramètres : puissance (26 à 55 dBm), azimut (0° à 359°), tilt (-15° à 0°)
- nombre d'émetteurs (TRX) : 1 à 7 selon trafic à assurer

Positionnement d'antennes pour les réseaux radio-mobiles

Déterminer

- un sous ensemble de sites parmi les sites candidats,
- pour chaque site sélectionné, le nombre et le type d'antennes,
- pour chaque antenne, la valeur de chacun de ses 3 paramètres (puiss, tilt et azimut).

Les contraintes impératives

- *Couverture* : tous les STP doivent être couverts par une antenne,
- *Connexité locale* : toute cellule constitue une seule composante connexe (en V8),
- *'Hand-over'* : toute cellule doit avoir des points de recouvrement avec ses voisines.

Objectifs

- minimiser le nombre de sites sélectionnés,
- minimiser les interférences générées par les antennes,
- maximiser le trafic supporté par le réseau,
- maximiser le rendement des émetteurs de chaque antenne.

Remarques :

- très grande combinatoire pour le choix d'un positionnement réalisable
- grande complexité de calcul pour vérifier les contraintes et pour évaluer les objectifs
- importante demande de ressource mémoire (200 à 500 MO de données par jeux)

Positionnement d'antennes pour les réseaux radio-mobiles

Contexte

- quelques études pour les réseaux micro cellulaires (indoor), rien pour les réseaux cellulaires de grande taille
- Projet européen ESPRIT 4 ARNO (Algorithms for Radio Network Optimisation)
- Méthodes heuristiques étudiées :
  - recuit simulé, tabou, génétique, réseaux de neurone
  - heuristiques spécifiques

Une approche heuristique fondée sur la méthode tabou (Vasquez & Hao 00a)

- Pré-traitement pour réduire par une heuristique la combinatoire
- Optimisation par **méthode tabou** pour rechercher des solutions réalisables
- Post-optimisation par raffinement local pour améliorer la solution

Remarques :

- Une méta-heuristique seule n'est pas suffisante pour aborder le problème

Positionnement d'antennes : résultats

Caractéristiques des réseaux

Réseaux	Sites potent.	Min. sites	Min. cellules	STP
Construction				
Autoroute	250	25	75	29954
Rural	320	22	65	72295
Petite zone urbaine	568	24	70	17393
Ville à grand trafic	244	21	61	48512
Extension				
Autoroute	250	25	75	29954
Rural	320	47	140	80854
Petite zone urbaine	568	63	189	42492
Ville à grand trafic	244	113	337	48512

Positionnement d'antennes : résultats

Un réseau urbain : 50 km x 46 km

- 568 sites potentiels
- 56792 points de mesure du signal radio
- 17393 points de service (STP)
- 6652 points de trafic (TTP) pour 2988.08 'erlang'

Contraintes :

- toutes les contraintes satisfaites : couverture, connexité locale et hand-over.

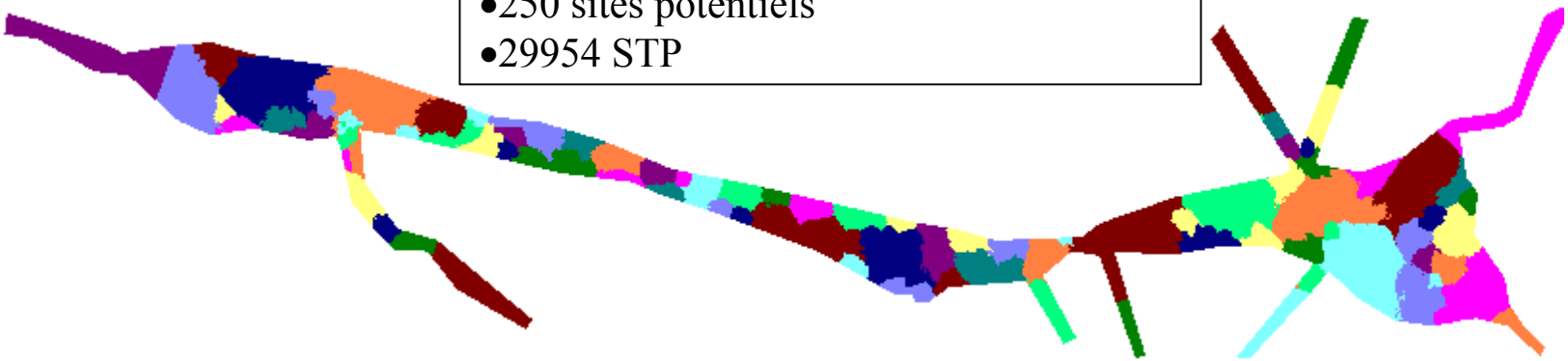
Objectifs :

- nombre de sites / antennes : 34 / 52 (35 direct. petite ouverture, 17 direct. large ouverture)
- interférences : niveau très bas
- trafic assuré : 85%
- rendement des émetteurs : très bon

Remarque : les approches par pénalités n'ont pas trouvé de solution réalisable.

Positionnement d'antennes : résultats

Un réseau autoroute : 39km x 168km
- 250 sites potentiels
- 29954 STP



Contraintes :

- toutes les contraintes satisfaites : couverture, connexité locale et hand-over.

Objectifs :

- nombre de sites/antennes : 58 / 103
- 1 omni-directionnelle, 67 direct. petite ouverture, 35 direct. large ouverture
- interférences : niveau très bas
- trafic assuré : moyen
- rendement des émetteurs : très bon

Planification journalière de prises de vues d'un satellite

Le problème

étant donné :

- un ensemble P de photographies (mono ou stéréo) à planifier pour le jour suivant ;
- une pondération (l'agrégation de certains critères) pour chaque photo ;
- les différentes possibilités de réalisation de chaque photographie :
  - trois pour les mono (une des trois caméras utilisables - avant, milieu et arrière),
  - une seule pos. pour les stéréo (à réaliser conjoint. par les caméras avant et arrière)
- un ensemble de contraintes binaires et ternaires à respecter impérativement (non chevauchement et respect des temps de transitions entre prises de vues...)
- une contrainte de mémoire : non dépassement de la capacité en mémoire d'une caméra par la somme des photos réalisées sur la caméra.

déterminer un sous-ensemble $P' \subseteq P$ tel que :

- la somme des pondérations associées à l'ensemble des photos de P' soit maximale toutes les contraintes soient respectées.

Remarque :

peut être formalisé comme un un sac-à-dos en variable 0/1 sous contraintes logiques

Planification de prises de vues

Méthodes de résolution

- algorithme exact (Verfaillie et al. 96)
- algorithmes gloutons (Agnèse et al. 95)
- algorithme tabou (Agnèse et al. 95)

Un algorithme tabou récent (Vasquez & Hao 00b)

- une formalisation du pb. en sac-à-dos 0/1 sous contraintes (au lieu de CSP)
- un algorithme tabou intégrant :
  - une technique de relaxation de la contrainte de mémoire
  - un voisinage efficace
  - une technique d'évaluation incrémentale rapide
  - une gestion dynam. de la liste tabou fondée sur les fréqu. de mouvements

Calcul des bornes supérieures (Vasquez & Hao 00c)

- Programmation dynamique et tabou (partition de graphes)

Planification de prises de vues

Jeux de test (définis par le CNES, disponibles sur l'internet)

- deux types
  - sans contrainte de mémoire (optimum connu)
  - avec contrainte de mémoire (optimum non connu)
- de grande taille, jusqu'à
  - 900 variables entières
  - 17 000 contraintes (binaires, ternaires et n-aines)

Résultats

- jeux sans contrainte de mémoire : solution optimale en moins de deux minutes
- jeux avec contrainte de mémoire :
  - quelques secondes pour atteindre les meilleures solutions connues
  - amélioration des meilleures solutions connues en quelques minutes

Conclusions

- Les métaheuristique
  - founit un nouveau *cadre* de résolution heuristique très général pour aborder des problèmes combinatoires (et continus) qu'on ne sait pas traiter autrement
  - constituent une classe de méthodes complémentaires à d'autres méthodes de résolutions plus conventionnelles (exactes, heuristiques de construction…)
  - peuvent être hybridées (avec raison) avec ces dernières
- Une bonne pratique nécessite l'adaptation « réfléchie » d'une métaheuristique au problème traité :
  - modélisation adéquate du problème en terme de contraintes et d'objectifs
  - intégration des connaissances spécifiques du problème
    - représentation de l'espace de recherche
    - opérateurs sémantiquement spécifiques (voisinage, croisement, mutation…)
    - traitement des contraintes
    - fonction d'évaluation discriminante
    - des structures de données efficaces
- *Limite et danger* :
  - les métaheuristiques ne sont pas une panacée
  - une pratique non-disciplinée conduit à des pertes du temps (et de l'argent)
  - la pratique reste encore trop personnelle (artisanal)

Quelques références générales

- Stochostoc Local Search: Fondations and Applications, H.H. Hoos & T. Stützle, Morgan Kaufmann, 2005.
- Métaheuristiques pour l'optimisation difficile, J. Dréo et al. 2003.
- Handbook of Applied Optimization, P.M. Pardalos & M.G.C. Resende, Oxford University Press, 2002.
- Modern Heuristic Techniques for Combinatorial Problems, C.R. Reeves (Ed.), McGraw-Hill, 1995.
- New Ideas in Optimization, D.Corne, M. Dorigo et F. Glover (Ed.), McGraw-Hill, 1999.
- Tabu Search, F. Glover & M. Laguna, Kluwer Academic Publishers, 1997.
- Simulated Annealing and Boltzmann Machines, E. Aarts & J. Korst, Johne Wiley & Sons, 1989.