

Recherche Locale Guidée pour la Coloration de Graphe

Daniel Porumbel, Jin Kao Hao, Pascale Kuntz

November 8, 2008

Problème de coloration:

Objectif K -coloration (graphe G)

Problème decision: Décider s'il y a une coloration avec K couleurs (K fixé) tel qu'il n'y ait pas deux sommets adjacents de même couleur

Problème optimisation: Chercher la coloration qui minimise le nombre de sommets adjacents de même couleur

Applications

Allocation de fréquences dans les réseaux mobiles, allocation des registres pour les compilateurs, problèmes d'ordonnancement, etc.

Problème de coloration:

Objectif K -coloration (graphe G)

Problème decision: Décider s'il y a une coloration avec K couleurs (K fixé) tel qu'il n'y ait pas deux sommets adjacents de même couleur

Problème optimisation: Chercher la coloration qui minimise le nombre de sommets adjacents de même couleur

Applications

Allocation de fréquences dans les réseaux mobiles, allocation des registres pour les compilateurs, problèmes d'ordonnancement, etc.

Problème de coloration:

Objectif K -coloration (graphe G)

Problème decision: Décider s'il y a une coloration avec K couleurs (K fixé) tel qu'il n'y ait pas deux sommets adjacents de même couleur

Problème optimisation: Chercher la coloration qui minimise le nombre de sommets adjacents de même couleur

Applications

Allocation de fréquences dans les réseaux mobiles, allocation des registres pour les compilateurs, problèmes d'ordonnancement, etc.

Difficulté et Complexité

- ▶ K-Coloration : **NP complet** pour $K \geq 3$
- ▶ Coloration : Le problème général est de déterminer le nombre minimal de couleurs K tel que G est/soit K -colorable
 - ▶ **NP complet**
 - ▶ **Non-approximable**: il n'admet pas d'algorithme d'approximation avec un facteur constant (sauf si $NP=P$)
- ▶ A partir de 100 sommets, les algorithmes exacts ne peuvent rien faire

Méthodes exactes et heuristiques

- ▶ Énumération complète de solutions potentielles:
 - ▶ jusqu'à 100 sommets au maximum
- ▶ Les heuristiques testent une partie de l'espace:
 - ▶ recherches locales: recuit simulé, (recherche Tabou) [Hertz et Al, Variable Search Space Algorithm for Graph Coloring, 2003], [Zufferey et. al, A Graph Coloring Heuristic Using Partial Solutions and a Reactive Tabu Scheme, 2007]
 - ▶ algorithmes avec des populations de colorations—i.e. génétiques, et hybrides, etc. [Fleurent et. Al, Genetic and hybrid algorithms for graph coloring, 1996], [Hao et Al, Hybrid Evolutionary Algorithms for Graph Coloring, 1999][Galinier et Al, An adaptive memory algorithm for graph coloring,2004]
 - ▶ construction incrementale (pas trop performante)

Principe de recherche locale

- ▶ Chaque coloration = un point dans un espace de recherche
- ▶ Les points de l'espace sont évalués par *la fonction objectif*:
 - ▶ Le nombre de conflits = le nombre d'arêtes avec les deux extrémités de la même couleur

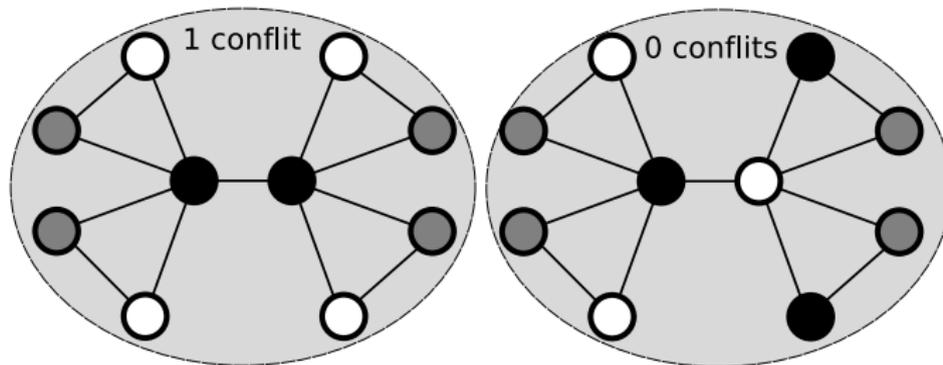
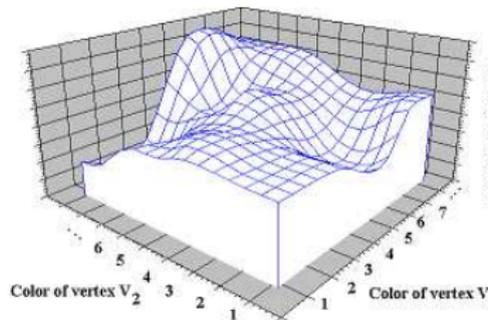


Figure: Exemples de colorations avec des évaluations différentes

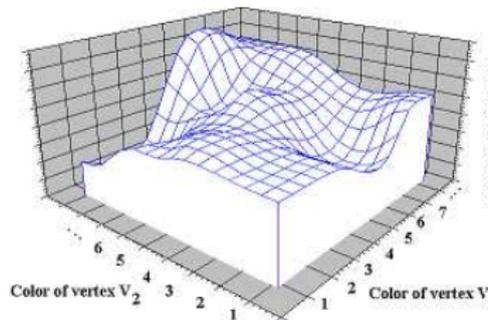
Recherche Locale Tabou

- ▶ La recherche locale passe d'une coloration à l'autre en changeant une couleur
- ▶ Descente simple:
 - ▶ Elle fait toujours le meilleur changement de couleur possible
 - ▶ On tombe toujours dans un optimum local
- ▶ Descente Tabou: Descent simple +
 - ▶ on interdit de (re-)visiter des colorations récemment visitées (pendant les dernières T itérations)
 - ▶ on peut monter s'il n'y a pas d'autres alternatives



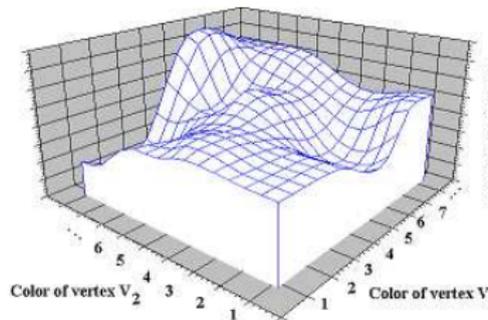
Recherche Locale Tabou

- ▶ La recherche locale passe d'une coloration à l'autre en changeant une couleur
- ▶ Descente simple:
 - ▶ Elle fait toujours le meilleur changement de couleur possible
 - ▶ On tombe toujours dans un optimum local
- ▶ Descente Tabou: Descent simple +
 - ▶ on interdit de (re-)visiter des colorations récemment visitées (pendant les dernières T_l itérations)
 - ▶ on peut monter s'il n'y a pas d'autres alternatives



Recherche Locale Tabou

- ▶ La recherche locale passe d'une coloration à l'autre en changeant une couleur
- ▶ Descente simple:
 - ▶ Elle fait toujours le meilleur changement de couleur possible
 - ▶ On tombe toujours dans un optimum local
- ▶ Descente Tabou: Descent simple +
 - ▶ on interdit de (re-)visiter des colorations récemment visitées (pendant les dernières T_l itérations)
 - ▶ on peut monter s'il n'y a pas d'autres alternatives

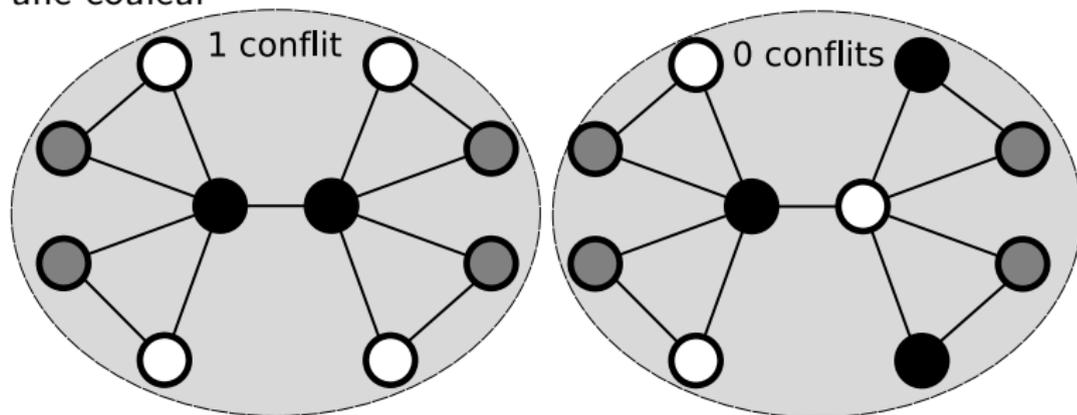


Vers une cartographie de l'espace

- ▶ Toute recherche locale (y compris Tabou) énumère UNE PARTIE de l'espace:
 - ▶ Il n'y a pas de vrais guidage.
 - ▶ Elle peut se bloquer, cycler dans quelques régions, répéter, etc.
- ▶ Il n'y a pas d'algorithme pour énumérer tout l'espace, mais on va essayer d'énumérer les régions!
 - ▶ On ajoute un processus qui enregistre la trace de la recherche
 - ▶ On fait une analyse de l'espace pour définir des régions

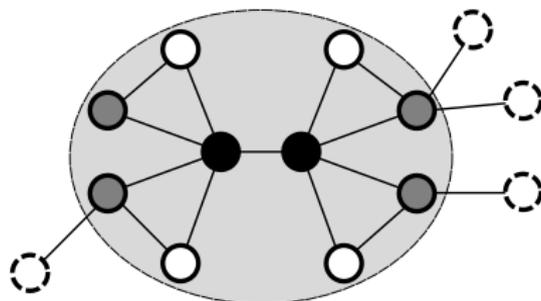
Exemple de situation difficile

- ▶ La recherche passe d'une coloration à l'autre en changeant une couleur



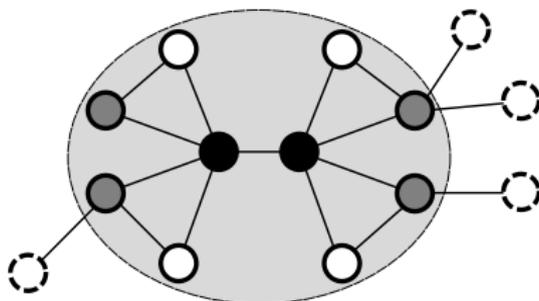
- ▶ Comment peut-elle passer de gauche à droite ?

Un exemple encore plus difficile



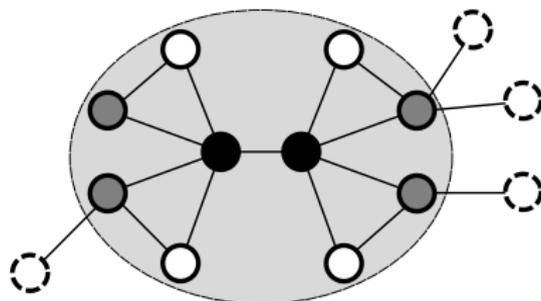
- ▶ On peut avoir des milliers de sommets périphériques (à l'extérieur de zone grise) \implies Beaucoup de colorations équivalentes (i.e. des colorations très proches)
- ▶ Une recherche classique peut changer tout le temps leurs couleurs, mais en vain!
- ▶ On utilise une distance entre colorations: le nombre de sommets qui doivent changer leurs classes

Un exemple encore plus difficile



- ▶ On peut avoir des milliers de sommets périphériques (à l'extérieur de zone grise) \implies Beaucoup de colorations équivalentes (i.e. des colorations très proches)
- ▶ Une recherche classique peut changer tout le temps leurs couleurs, mais en vain!
- ▶ On utilise une distance entre colorations: le nombre de sommets qui doivent changer leurs classes

Un exemple encore plus difficile



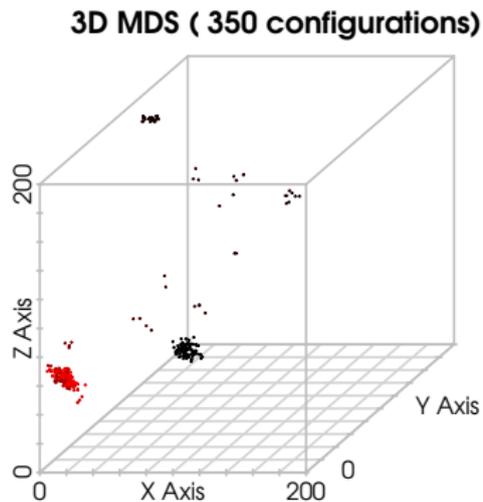
- ▶ On peut avoir des milliers de sommets périphériques (à l'extérieur de zone grise) \implies Beaucoup de colorations équivalentes (i.e. des colorations très proches)
- ▶ Une recherche classique peut changer tout le temps leurs couleurs, mais en vain!
- ▶ On utilise une distance entre colorations: le nombre de sommets qui doivent changer leurs classes

Distance (de transfert) entre colorations

- ▶ Une coloration = une partition de l'ensemble de sommets
- ▶ Étant données deux partitions P et Q , la distance de transfert: le *nombre minimum de transferts* d'un élément d'une classe dans une autre nécessaires pour transformer P en Q . [Irène Charon, Lucile Denoeud et Olivier Hudry, Maximum de la distance de transfert à une partition donnée, 2008]
- ▶ Cette distance représente le nombre minimal de changements de couleurs nécessaires pour arriver de coloration P à Q

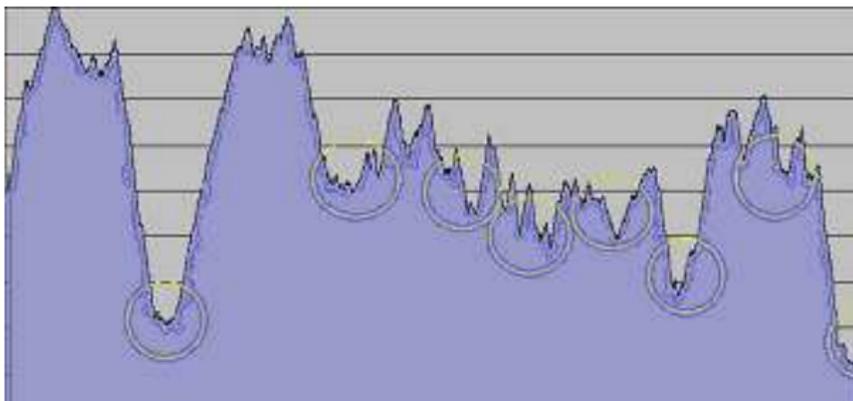
Arrangement des minima locaux

- ▶ Cette figure présente 350 minimums locaux dans l'espace de recherche
- ▶ On l'a générée avec une procédure "Multidimensional scaling"
- ▶ Les distances entre les points 3D tentent de préserver au mieux la distance entre les configurations correspondantes dans l'espace $|V|$ -dimensionnel



- ▶ On remarque les clusters : ils ont un rayon d'environ 10
 - ▶ Notre graphe a 250 sommets \Rightarrow distance $\in \{0, 1, \dots, 250\}$

Un exemple de profil d'espace de recherche



- ▶ On ne peut pas stocker tous les points de l'espace
- ▶ C'est possible de stocker les 8 sphères dessinées (correspondent à des optimums locaux)
- ▶ La recherche ne doit pas rester tout le temps dans un seul cercle.
 - ▶ De cette façon, on est sûr de trouver la meilleure solution

Recherche locale Tabou + un processus d'apprentissage

- ▶ On considère l'espace de recherche divisé en sphères:
- ▶ La sphère d'une coloration C est l'ensemble des colorations situées à moins de $10\%|V|$ distance de C
- ▶ L'algorithme tourne comme d'habitude mais il enregistre chaque sphère qu'il visite

Recherche Tabou + apprentissage + utilisation d'informations apprises

- ▶ Le processus d'apprentissage a enregistré toutes les sphères visitées
- ▶ Dès que la recherche tombe dans la sphère d'une coloration déjà visitée
⇒
- ▶ On déclenche une phase de diversification, i.e. une phase avec des changements de couleurs plus diverses
 - ▶ Cette phase va tourner la recherche à l'extérieur de la région
 - ▶ On garantit que la recherche explore toujours de nouvelles pistes, elle ne répète rien !

Avantages et désavantages

- +++ Le processus de recherche explore de nouvelles régions tout le temps
- +++ Pour les graphes plus petits, c'est vraiment possible d'énumérer chaque sphère: le nombre de sphères peut varier de $K^{|V|}$ (minimum radius) à 1 (maximum radius)
- Une sphère prometteuse peut être explorée que une seule fois, peut-être cela n'est pas suffisant

Avantages et désavantages

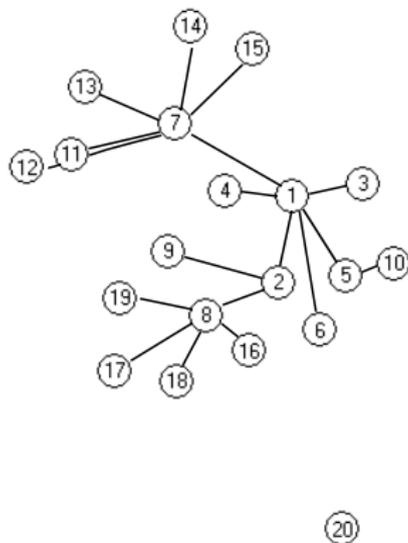
- +++ Le processus de recherche explore de nouvelles régions tout le temps
- +++ Pour les graphes plus petits, c'est vraiment possible d'énumérer chaque sphère: le nombre de sphères peut varier de $K^{|V|}$ (minimum radius) à 1 (maximum radius)
- Une sphère prometteuse peut être explorée que une seule fois, peut-être cela n'est pas suffisant

Avantages et désavantages

- +++ Le processus de recherche explore de nouvelles régions tout le temps
- +++ Pour les graphes plus petits, c'est vraiment possible d'énumérer chaque sphère: le nombre de sphères peut varier de $K^{|V|}$ (minimum radius) à 1 (maximum radius)
- Une sphère prometteuse peut être explorée que une seule fois, peut-être cela n'est pas suffisant

Algorithme Intensification

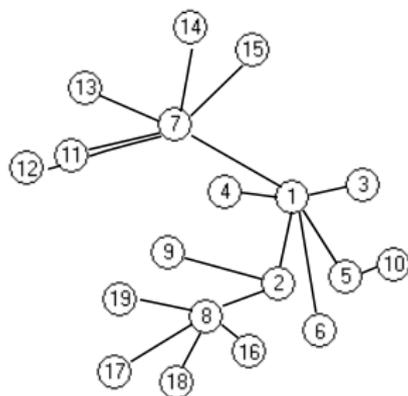
- ▶ On voit un exemple d'une évolution réelle, le point 20 est *la solution* (Multidimensional scaling en 2D)



- ▶ 1 est une sphère/coloration stockée par l'algorithme précédent (le point de départ)
- ▶ Les voisins d'un point X sont les colorations découvertes par une recherche arrêtée dès qu'elle sort de la sphère de X
- ▶ On explore l'espace avec un algorithme de parcours en largeur: les chiffres montrent l'ordre de visite

Algorithme Intensification

- ▶ On voit un exemple d'une évolution réelle, le point 20 est *la solution* (Multidimensional scaling en 2D)

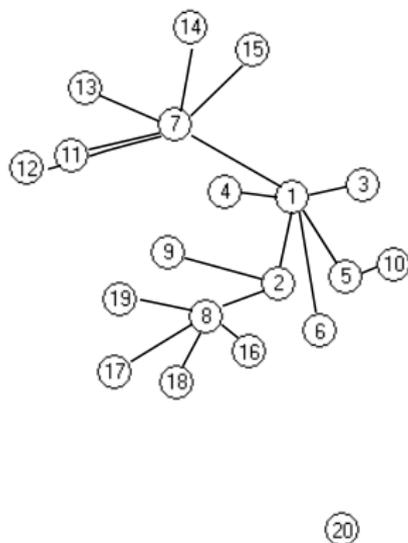


- ▶ 1 est une sphère/coloration stockée par l'algorithme précédent (le point de départ)
- ▶ Les voisins d'un point X sont les colorations découvertes par une recherche arrêtée dès qu'elle sort de la sphère de X
- ▶ On explore l'espace avec un algorithme de parcours en largeur: les chiffres montrent l'ordre de visite

20

Algorithme Intensification

- ▶ On voit un exemple d'une évolution réelle, le point 20 est *la solution* (Multidimensional scaling en 2D)



- ▶ 1 est une sphère/coloration stockée par l'algorithme précédent (le point de départ)
- ▶ Les voisins d'un point X sont les colorations découvertes par une recherche arrêtée dès qu'elle sort de la sphère de X
- ▶ On explore l'espace avec un algorithme de parcours en largeur: les chiffres montrent l'ordre de visite

Algorithme Intensification

Create *sorted queue* Q (with one element: the input config.)

While Q is **not** empty

1. $C_s = \text{FRONT}(Q)$
2. **Launch** TS starting from C_s
3. let C_c be the first configuration such that $d(C_s, C_c) > 10\%$ (i.e. this becomes a stopping condition in algorithm ??)
4. **if** $d(C_i, C_c) > 10\%$ **forall** $C_i \in Q$
 - ▶ $\text{PUSH}(C_c, Q)$ // insert it at the correct position
5. **if** a *stopping condition* is met
 - ▶ $\text{POP}(Q)$

Tableau de résultats

Graph	χ, K^*	TS	Notre algorithme	VSS 2008	PCol 2008	ACol 2008	MOR 1993	GH 1999	MMT 2008
<i>dsjc250.5</i>	?, 28	28	28	-	-	28	28	28	28
<i>dsjc500.1</i>	?, 12	13	12	12	12	12	12	-	12
<i>dsjc500.5</i>	?, 48	49	48	48	48	48	49	48	48
<i>dsjc500.9</i>	?, 126	127	126	126	126	126	??	-	126
<i>dsjc1000.1</i>	?, 20	21	20	20	20	20	21	20	20
<i>dsjc1000.5</i>	?, 83	88	85	88	88	84	88	83	83
<i>dsjc1000.9</i>	?, 224	225	223	224	225	224	226	224	226
<i>le450.25c</i>	25, 25	26	25	26	27	26	25	26	25
<i>le450.25d</i>	25, 25	26	25	26	27	26	25	26	25
<i>flat300.28</i>	28, 32	30	28	29	28	31	31	31	31
<i>flat1000.76</i>	76, 82	87	85	87	87	84	89	83	82
<i>r1000.1c</i>	?, 98	98	98	-	98	-	98	-	98

Table: Comparaison avec les meilleurs algorithmes

Résultats en bref:

1. On a trouvé une nouvelle coloration (DSJC1000.9, $K=223$)
2. On trouve presque toujours les meilleures colorations connues
3. Si on restreint la comparaison avec les recherches locales publiées, on obtient les meilleurs résultats sur tous les graphes

Conclusion

- ▶ Nous avons présenté les idées principales d'une recherche qui mémorise sa trace
 - ▶ Elle est garantie d'explorer de nouvelles régions tout le temps
 - ▶ Elle peut arriver à énumérer toutes les régions de l'espace si elle a le temps
 - ▶ Le deuxième algorithme explore le voisinage des certaines colorations prometteuses
- ⇒
- ▶ Les résultats sont très bons, le principe peut s'appliquer pour n'importe quel problème où on peut définir une distance