

Algorithmes robustes des graphes, Ecole Jeunes Chercheurs Montpellier 2005

Michel Habib,
LIRMM Montpellier

20 juin 2005

Plan

- 1 Introduction à l'algorithmique discrète
- 2 Algorithmes robustes et certificats
 - Pragmatique
 - Complexité
 - Conséquences
- 3 Reconnaissance des graphes triangulés
 - Rappels
- 4 Autres exemples d'algorithmes robustes
 - Arbres de poids minimum
 - Plus courts chemins
- 5 Perspectives

- 1 Introduction à l'algorithmique discrète
- 2 Algorithmes robustes et certificats
 - Pragmatique
 - Complexité
 - Conséquences
- 3 Reconnaissance des graphes triangulés
 - Rappels
- 4 Autres exemples d'algorithmes robustes
 - Arbres de poids minimum
 - Plus courts chemins
- 5 Perspectives

Un algorithme très simple de V. Chvátal

Données: σ une permutation de $[1, n]$

Résultat: une permutation σ' telle que $\sigma'(1) = 1$

while $\sigma(1) \neq 1$ **do**

 └ Renverser l'ordre des $\sigma(1)$ premiers éléments

Un algorithme très simple de V. Chvátal

Considérons σ une permutation de $[1,7]$

6 3 1 7 2 5 4

5 2 7 1 3 6 4

3 1 7 2 5 6 4

7 1 3 2 5 6 4

4 6 5 2 3 1 7

2 5 6 4 3 1 7

5 2 6 4 3 1 7

3 4 6 2 5 1 7

6 4 3 2 5 1 7

1 5 2 3 4 6 7 OUF !

Un algorithme très simple de V. Chvátal

Terminaison

Il est facile de montrer que l'algorithme précédent termine en un nombre fini d'étapes.

Un algorithme très simple de V. Chvátal

Terminaison

Il est facile de montrer que l'algorithme précédent termine en un nombre fini d'étapes.

Complexité

On peut borner par $O(2^n)$

Un algorithme très simple de V. Chvátal

Terminaison

Il est facile de montrer que l'algorithme précédent termine en un nombre fini d'étapes.

Complexité

On peut borner par $O(2^n)$

Conjecture de Chvátal

Mais il est possible que $O(n^2)$ ou $O(n)$ soit la solution ?

Moralité

- Il n'est pas si facile d'analyser la complexité d'un algorithme possédant une seule instruction.
- Ces opérations de retournement interviennent en bioinformatique (tri par inversion de permutations).

- 1 Introduction à l'algorithmique discrète
- 2 Algorithmes robustes et certificats**
 - Pragmatique
 - Complexité
 - Conséquences
- 3 Reconnaissance des graphes triangulés
 - Rappels
- 4 Autres exemples d'algorithmes robustes
 - Arbres de poids minimum
 - Plus courts chemins
- 5 Perspectives

Un exemple motivant

Kurt Mehlhorn et son équipe de la librairie LEDA proposaient un programme de planarité de graphes qui vérifiait :

Un exemple motivant

Kurt Mehlhorn et son équipe de la librairie LEDA proposaient un programme de planarité de graphes qui vérifiait :

- Réponse **OUI** = un dessin planaire
- Réponse **NON** = Le graphe n'est pas planaire

Un exemple motivant

Kurt Mehlhorn et son équipe de la librairie LEDA proposaient un programme de planarité de graphes qui vérifiait :

- Réponse **OUI** = un dessin planaire
- Réponse **NON** = Le graphe n'est pas planaire

Il a fallu deux ans pour s'apercevoir que ce programme était faux !

Un exemple motivant

Il aurait fallu un programme qui :

Un exemple motivant

Il aurait fallu un programme qui :

- Réponse **OUI** = "un dessin planaire"
- Réponse **NON** = " une obstruction $K_{3,3}$ ou K_5 "

Un exemple motivant

Il aurait fallu un programme qui :

- Réponse **OUI** = "un dessin planaire"
- Réponse **NON** = " une obstruction $K_{3,3}$ ou K_5 "

Basé sur le théorème de Kuratowski, fournissant un certificat vérifiable en $O(n + m)$ dans les deux cas.

Comment formaliser cette idée ?

Il faut interpréter ces idées de certificat et de vérification.
On trouve dans la littérature deux notions voisines

Comment formaliser cette idée ?

Il faut interpréter ces idées de certificat et de vérification.
On trouve dans la littérature deux notions voisines

La première pragmatique

Les algorithmes munis de certificats "faciles" à vérifier.
Facile = polynomial.

Comment formaliser cette idée ?

Il faut interpréter ces idées de certificat et de vérification.
On trouve dans la littérature deux notions voisines

La première pragmatique

Les algorithmes munis de certificats "faciles" à vérifier.
Facile = polynomial.

La deuxième provenant de la théorie de la complexité algorithmique

Les (Existentially Provable) EP théorèmes, J. Edmonds 1990.
Les algorithmes **robustes**, J. Spinrad 2002.

Principales références

- K. Cameron, J. Edmonds, Existentially polytime theorems, Dimacs Series in DMTCS, (1990), 83-100.
- D. Kratsch, R.M. Connell, K. Mehlhorn, J. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs, SODA 2003.
- V. Raghavan, J. Spinrad, Robust algorithms for restricted domains, J. of Algorithms 48 (2003) 160-172.
- J. Spinrad, Efficient graph representations, Fields Institute Monographs, 2003.
- H. Wasserman, M. Blum, Software reliability via run-time Result checking, JACM 44 (1997) 826-849.

Certificat versus Preuve

Tout ce que l'on peut dire, c'est qu'à chaque usage du programme on peut vérifier le résultat du programme.

Certificat versus Preuve

Tout ce que l'on peut dire, c'est qu'à chaque usage du programme on peut vérifier le résultat du programme.

Sinon il faudrait :

- Prouver l'algorithme (Méthode des invariants)
- Prouver l'adéquation Programme – Algorithme
- Prouver le programme (Structures de données ...)
- Faire confiance au compilateur, au système ...

Complexité

Dans le cadre des problèmes de décision.

Complexité

Dans le cadre des problèmes de décision.

La symétrie des réponses OUI–NON nous restreint à $NP \cap co - NP$

Complexité

Dans le cadre des problèmes de décision.

La symétrie des réponses OUI–NON nous restreint à $NP \cap co - NP$

Comment certifier que la valeur d'une solution donnée par une heuristique sur un problème NP-difficile est inférieure à K fois l'optimum ?

Bonnes caractérisations

NP la classe des problèmes de décision admettant un certificat polynomial lorsque la réponse est OUI.

Bonnes caractérisations

NP la classe des problèmes de décision admettant un certificat polynomial lorsque la réponse est OUI.

$NP \cap co - NP$ la classe des bonnes caractérisations
(À l'origine des théories sur la complexité, J. Edmonds 1965)

Bonnes caractérisations

NP la classe des problèmes de décision admettant un certificat polynomial lorsque la réponse est OUI.

$NP \cap co - NP$ la classe des bonnes caractérisations
(À l'origine des théories sur la complexité, J. Edmonds 1965)

Gardons comme exemples les problèmes de graphes car cela permet beaucoup de choses (cf. les théorèmes de caractérisation de P et NP de Fagin).

Fameuse conjecture

La conjecture $P = NP \cap co - NP$? a eu des conséquences très importantes.

Fameuse conjecture

La conjecture $P = NP \cap co - NP ?$ a eu des conséquences très importantes.

- Programmation linéaire en nombre réels (Kachian, 1979)
- Test de primalité (2002)
- Graphes parfaits (2003)
- Jeux de Parité ?
- Transversal minimal ?

EP théorèmes, J. Edmonds 1990

Un théorème EP (Existentially Polytime) est un théorème dont chaque condition est vérifiable polynomialement.

EP théorèmes, J. Edmonds 1990

Un théorème EP (Existentially Polytime) est un théorème dont chaque condition est vérifiable polynomialement.

Exemples :

- une bonne caractérisation
- un graphe n'est pas parfait ss'il contient un trou (cycle sans corde) impair ou son complémentaire.

EP théorèmes

Sans vraiment l'écrire explicitement, J. Edmonds pense qu'un tel théorème implique l'existence d'un algorithme polynomial (au moins ceux du type $NP \cap co - NP$).

Exemple : La reconnaissance des graphes parfaits (2003).

Algorithmes robustes, J. Spinrad 2002

Dans le cadre d'un pb de décision ou d'optimisation NP-complet en général (ex : coloration), si l'on travaille sur une classe C de graphes particulière, un algorithme polynomial est dit robuste s'il vérifie :

Algorithmes robustes, J. Spinrad 2002

- ① Si la donnée est dans la classe C , l'algorithme donne la bonne réponse.
- ② Si la donnée n'est pas dans la classe :
 - Soit l'algorithme donne la bonne réponse
 - Soit l'algorithme répond que la donnée n'est pas dans la classe et exhibe un certificat.

Algorithmes robustes, J. Spinrad 2002

Exemples :

- Trivial : calcul d'une 2-coloration (vérification qu'un graphe est biparti)
- Un algorithme facile à vérifier est robuste.

Algorithmes robustes, J. Spinrad 2002

Exemples :

- Trivial : calcul d'une 2-coloration (vérification qu'un graphe est biparti)
- Un algorithme facile à vérifier est robuste.

Paradoxe

Certains algorithmes robustes sont plus rapides que l'algorithme de reconnaissance de la classe C .

Algorithmes robustes, J. Spinrad 2002

La reconnaissance d'un graphe de comparabilité (graphe admettant une orientation transitive).

Algorithmes robustes, J. Spinrad 2002

La reconnaissance d'un graphe de comparabilité (graphe admettant une orientation transitive).

Calcul d'une orientation transitive $O(n + m)$

Algorithmes robustes, J. Spinrad 2002

La reconnaissance d'un graphe de comparabilité (graphe admettant une orientation transitive).

Calcul d'une orientation transitive $O(n + m)$

Vérifier qu'une l'orientation est bien transitive est un pb équivalent au calcul d'un produit de matrices.

Algorithmes robustes, J. Spinrad 2002

Un exemple intéressant

Un algorithme robuste linéaire pour le calcul d'une clique de cardinal max dans un graphe de comparabilité :

Algorithmes robustes, J. Spinrad 2002

Un exemple intéressant

Un algorithme robuste linéaire pour le calcul d'une clique de cardinal max dans un graphe de comparabilité :

Principes

- 1 Calcul d'une orientation transitive en $O(n + m)$
- 2 Calcul du plus long chemin en $O(n + m)$
- 3 Certificat : le plus long chemin. Vérifiable en $O(n + m)$.

Algorithmes robustes, J. Spinrad 2002

2-coloration

Réponse OUI : une partition des sommets en deux stables

$O(n + m)$

Réponse NON : un cycle impair $O(n)$

Algorithmes robustes, J. Spinrad 2002

2-coloration

Réponse OUI : une partition des sommets en deux stables

$O(n + m)$

Réponse NON : un cycle impair $O(n)$

Clique max d'un graphe de comparabilité

Lorsque l'algorithme donne l'optimum la vérification se fait en $O(n + m)$, dans le cas contraire il faut vérifier que l'orientation fournie est bien licite. Le certificat est donc basé sur l'algorithme lui-même.

Cas défavorables

Lorsque le seul certificat est l'algorithme lui-même (certificats garantis par la théorie de la complexité).

Cas défavorables

Lorsque le seul certificat est l'algorithme lui-même (certificats garantis par la théorie de la complexité).

Cas favorables

Les deux certificats des réponses OUI et NON sont vérifiables de manière indépendante de l'algorithme.

Cas très favorables

Une nouvelle classe : les algorithmes **très faciles à vérifier** : Le certificat doit être vérifiable avec une complexité non supérieure à celle de l'algorithme

Cas très favorables

Une nouvelle classe : les algorithmes **très faciles à vérifier** : Le certificat doit être vérifiable avec une complexité non supérieure à celle de l'algorithme

Modèles

- Graphes 2-colorables (bipartis) ($O(n + m)$, $O(n)$).
- Cographes ou graphes sans P_4 ($O(n + m)$, $O(1)$).
- Graphes d'intervalles, graphes de permutations ($O(n + m)$, $O(n)$). (Kratsch et al 2003)

Robustesse

Certains problèmes n'admettent pas d'algorithmes robustes (à moins que $P = NP$).

Il reste de nombreuses questions ouvertes :

Robustesse

Certains problèmes n'admettent pas d'algorithmes robustes (à moins que $P = NP$).

Il reste de nombreuses questions ouvertes :

Problèmes ouverts

Clique maximale sur les graphes de visibilité ?

Un algorithme robuste sur certaines instances polynomiales de SAT ?

...

EP théorème et algorithmes robustes

Tout algorithme robuste correspond à un EP théorème. L'existence d'un EP théorème peut servir de guide à la recherche d'un algorithme robuste.

Une nouvelle manière de faire de l'algorithmique

Appliquons ces idées à quelques problèmes classiques (exemple recherche dans un tableau).

Une nouvelle manière de faire de l'algorithmique

Appliquons ces idées à quelques problèmes classiques (exemple recherche dans un tableau).

Problèmes très faciles à tester (complexité du test inférieure à la complexité de l'algorithme).

Une nouvelle manière de faire de l'algorithmique

Appliquons ces idées à quelques problèmes classiques (exemple recherche dans un tableau).

Problèmes très faciles à tester (complexité du test inférieure à la complexité de l'algorithme).

Problèmes difficiles à tester. On peut montrer que le test nécessite au moins autant de temps que l'algorithme lui-même.

Une nouvelle manière de faire de l'algorithmique

Appliquons ces idées à quelques problèmes classiques (exemple recherche dans un tableau).

Problèmes très faciles à tester (complexité du test inférieure à la complexité de l'algorithme).

Problèmes difficiles à tester. On peut montrer que le test nécessite au moins autant de temps que l'algorithme lui-même.

Pour chaque algorithme produit, se poser la question : Existe-t-il un autre moyen que l'algo lui-même qui permette de valider le résultat ?

- 1 Introduction à l'algorithmique discrète
- 2 Algorithmes robustes et certificats
 - Pragmatique
 - Complexité
 - Conséquences
- 3 Reconnaissance des graphes triangulés**
 - Rappels**
- 4 Autres exemples d'algorithmes robustes
 - Arbres de poids minimum
 - Plus courts chemins
- 5 Perspectives

Rappel des définitions

Graphe triangulé

Un graphe est triangulé si tout cycle de longueur ≥ 4 possède une corde.

Rappel des définitions

Graphe triangulé

Un graphe est triangulé si tout cycle de longueur ≥ 4 possède une corde.

Sommet simplicial

Un sommet est simplicial si son voisinage est une clique.

Rappel des définitions

Graphe triangulé

Un graphe est triangulé si tout cycle de longueur ≥ 4 possède une corde.

Sommet simplicial

Un sommet est simplicial si son voisinage est une clique.

Ordre d'élimination simplicial

$\sigma = [x_1 \dots x_i \dots x_n]$ est un ordre d'élimination simplicial si $\forall i, x_i$ est simplicial dans le sous-graphe $G_i = G[\{x_i \dots x_n\}]$

Caractérisations utiles

Caractérisation [Dirac 1961]

Un graphe est triangulé ss'il possède un ordre d'élimination simplicial.

Caractérisations utiles

Caractérisation [Dirac 1961]

Un graphe est triangulé ss'il possède un ordre d'élimination simplicial.

Caractérisation LexBFS [Rose, Tarjan et Lueker 1976]

Un graphe G est triangulé ssi tout ordre "inverse" LexBFS de G est simplicial.

LexBFS

Parcours en largeur lexicographique

Données: Un graphe $G = (V, E)$ et un sommet source s

Résultat: Un ordre total σ de V

- 1 Affecter l'étiquette \emptyset à chaque sommet
- 2 $label(s) \leftarrow \{n + 1\}$
- 3 **pour** $i \leftarrow 1$ à n **faire**
- 4 Choisir un sommet v d'étiquette lexicographique max.
- 5 $\sigma(i) \leftarrow v$
- 6 **pour chaque** *sommet non-numéroté* $w \in N(v)$ **faire**
- 7 $label(w) \leftarrow label(w) \cdot \{n - i\}$

Reconnaissance robuste

Etape 1

Calcul d'un parcours lexicographique en largeur

$LexBFS(x_n) = x_n, \dots, x_i, \dots, x_1$ en $O(n + m)$.

Reconnaissance robuste

Etape 1

Calcul d'un parcours lexicographique en largeur
 $\text{LexBFS}(x_n) = x_n, \dots, x_i, \dots, x_1$ en $O(n + m)$.

Etape 2

Vérification de la simplicialité.
En $O(n + m)$ dans le cas positif.

Le cas négatif

On a trouvé x_i possédant deux voisins a et b non adjacents

Le cas négatif

On a trouvé x_i possédant deux voisins a et b non adjacents

On considère les chemins $\mu(a)$ joignant a à x_n et $\mu(b)$ joignant b à x_n .

Chemins issus de l'arborescence sous-jacente au parcours LexBFS.
Soit z le premier sommet commun à ces chemins. Le cycle $[x_i, a, \dots, z, \dots, b, x_i]$ contient un cycle sans corde.

Le cas négatif

On a trouvé x_i possédant deux voisins a et b non adjacents

On considère les chemins $\mu(a)$ joignant a à x_n et $\mu(b)$ joignant b à x_n .

Chemins issus de l'arborescence sous-jacente au parcours LexBFS.
Soit z le premier sommet commun à ces chemins. Le cycle $[x_i, a, \dots, z, \dots, b, x_i]$ contient un cycle sans corde.

Complexité en $O(n)$.

- 1 Introduction à l'algorithmique discrète
- 2 Algorithmes robustes et certificats
 - Pragmatique
 - Complexité
 - Conséquences
- 3 Reconnaissance des graphes triangulés
 - Rappels
- 4 **Autres exemples d'algorithmes robustes**
 - Arbres de poids minimum
 - Plus courts chemins
- 5 Perspectives

Arbre de poids minimum

Un algorithme de construction en $O(n + m \log n)$

Arbre de poids minimum

Un algorithme de construction en $O(n + m \log n)$

Un algorithme de vérification en $O(n + m)$

Plus courts chemins

Un algorithme de type Dijkstra calcule en $O(n + m \log n)$ les plus courts chemins allant d'un sommet s à tous les autres.

Plus courts chemins

Un algorithme de type Dijkstra calcule en $O(n + m \log n)$ les plus courts chemins allant d'un sommet s à tous les autres.

Si l'on produit une arborescence des plus courts chemins T , alors il est facile de vérifier que c'est bien une arborescence des plus courts chemins en $O(n + m)$.

$$\forall e = (x, y) \in G - T, d_T(y) \geq d_T(x) + \omega(e)$$

Ordres totaux caractérisants

De nombreux algorithmes de graphes se ramènent à calculer un ordre caractéristique sur les sommets du graphe : une permutation des sommets.

Exemples : schémas simpliciaux, orientation transitive, (s,t) -ordre, graphe de permutation, permutation factorisante pour la décomposition modulaire. . . .

Ordres totaux caractérisants

De nombreux algorithmes de graphes se ramènent à calculer un ordre caractéristique sur les sommets du graphe : une permutation des sommets.

Exemples : schémas simpliciaux, orientation transitive, (s,t) -ordre, graphe de permutation, permutation factorisante pour la décomposition modulaire. . . .

Algorithme en 2 étapes

- 1 Calcul de l'ordre
- 2 Vérification des propriétés de l'ordre.

En général ces algorithmes sont faciles à vérifier.

- 1 Introduction à l'algorithmique discrète
- 2 Algorithmes robustes et certificats
 - Pragmatique
 - Complexité
 - Conséquences
- 3 Reconnaissance des graphes triangulés
 - Rappels
- 4 Autres exemples d'algorithmes robustes
 - Arbres de poids minimum
 - Plus courts chemins
- 5 Perspectives**

Algorithme

DIJKSTRA 1959

```
1  $d(s) \leftarrow 0$ 
2 forall  $v \in V \setminus \{s\}$  do
3    $d(v) \leftarrow \infty$ 
4  $U \leftarrow V$ 
5 while  $U \neq \emptyset$  do
6   Choisir  $u \in U$  tel que  $d(u)$  soit minimal
7    $U \leftarrow U \setminus \{u\}$ 
8   forall  $v \in V$  avec  $(u, v) \in E$  do
9      $d(v) \leftarrow \min\{d(v), d(u) + \omega(u, v)\}$ 
```

Perspectives

Vers des algorithmes linéaires : travaux récents de M. Thorup, A.V. Golberg, T. Haguerup (STACS 2004)

Perspectives

Vers des algorithmes linéaires : travaux récents de M. Thorup, A.V. Golberg, T. Haguerup (STACS 2004)

Relaxation de l'ordre total utilisé dans l'algorithme de Dijkstra.
Bonnes structures de données

Perspectives

Vers des algorithmes linéaires : travaux récents de M. Thorup, A.V. Golberg, T. Haguerup (STACS 2004)

Relaxation de l'ordre total utilisé dans l'algorithme de Dijkstra.
Bonnes structures de données

Construire un algorithme robuste de reconnaissance d'un path-graphe.
Il reste beaucoup de choses à trouver. Par exemple des algorithmes dynamiques et robustes.