Algorithmes faiblement exponentiels

Vangelis Th. Paschos

JFRO
20 novembre 2012

Résoudre à l'optimum un problème **NP**-difficile

Les méthodes les plus populaires (et les plus anciennes) :

- Enumération exhaustive
- Méthodes polyédrales
- *Branch & . . .*
- Programmation dynamique

Quelques questions

Quelques questions

- Quelle est la complexité de l'énumération exhaustive pour le problème du STABLE, du TSP, de la COLORATION, du SAC-A-DOS, de l'ENSEMBLE DOMINANT D'ARÊTES, des problèmes de suppression d'arêtes, . . . ?

Quelques questions

- Quelle est la complexité de l'énumération exhaustive pour le problème du STABLE, du TSP, de la COLORATION, du SAC-A-DOS, de l'ENSEMBLE DOMINANT D'ARÊTES, des problèmes de suppression d'arêtes, . . . ?
- D'un *branch & bound* pour la COUVERTURE ?

Quelques questions

- Quelle est la complexité de l'énumération exhaustive pour le problème du STABLE, du TSP, de la COLORATION, du SAC-A-DOS, de l'ENSEMBLE DOMINANT D'ARÊTES, des problèmes de suppression d'arêtes, . . . ?
- D'un *branch & bound* pour la COUVERTURE ?
- D'une méthode polyédrale pour l'arbre de Steiner ?

Quelques questions

- Quelle est la complexité de l'énumération exhaustive pour le problème du STABLE, du TSP, de la COLORATION, du SAC-A-DOS, de l'ENSEMBLE DOMINANT D'ARÊTES, des problèmes de suppression d'arêtes, . . . ?
- D'un *branch & bound* pour la COUVERTURE ?
- D'une méthode polyédrale pour l'arbre de Steiner ?
- Comment maîtriser l'explosion combinatoire de ces méthodes ?

Quelques questions

- Quelle est la complexité de l'énumération exhaustive pour le problème du STABLE, du TSP, de la COLORATION, du SAC-A-DOS, de l'ENSEMBLE DOMINANT D'ARÊTES, des problèmes de suppression d'arêtes, . . . ?
- D'un *branch & bound* pour la COUVERTURE ?
- D'une méthode polyédrale pour l'arbre de Steiner ?
- Comment maîtriser l'explosion combinatoire de ces méthodes ?
- Comment la quantifier le plus précisément possible ?

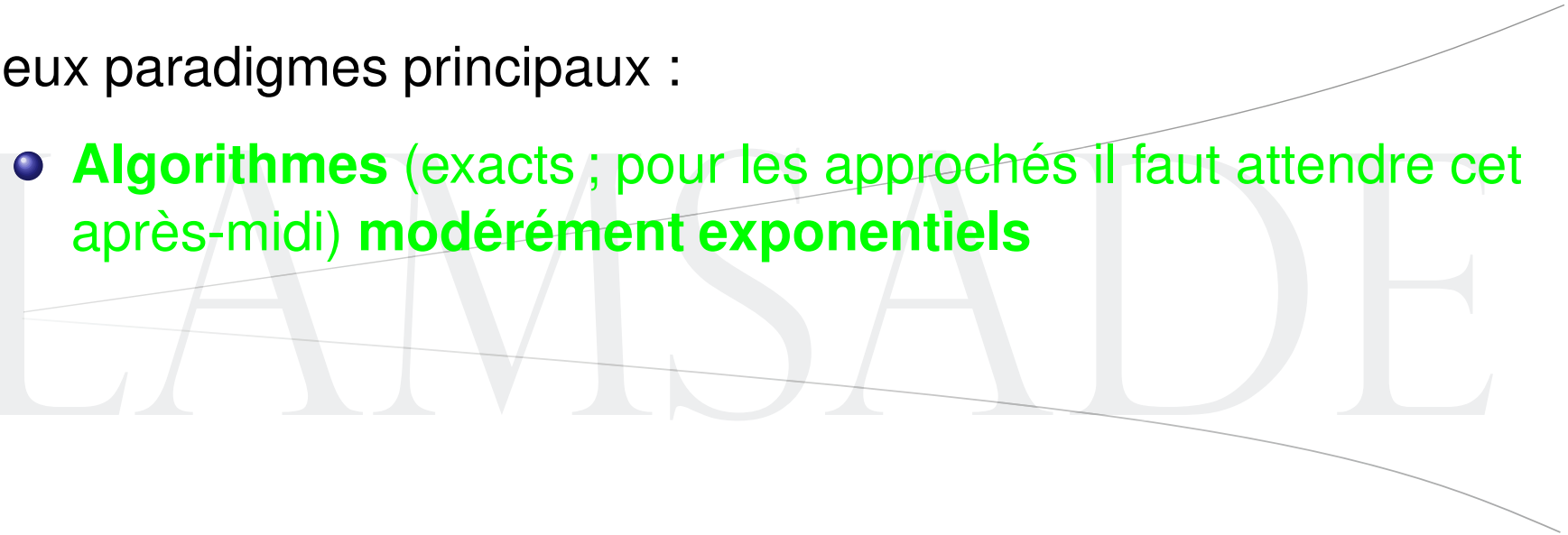
Et une réponse

Deux paradigmes principaux :

Et une réponse

Deux paradigmes principaux :

- **Algorithmes** (exacts ; pour les approchés il faut attendre cet après-midi) **modérément exponentiels**



Et une réponse

Deux paradigmes principaux :

- **Algorithmes** (exacts ; pour les approchés il faut attendre cet après-midi) **modérément exponentiels**
- Algorithmes paramétrés (pour cela il faut attendre à peu près une heure)

Et une réponse

Deux paradigmes principaux :

- **Algorithmes** (exacts ; pour les approchés il faut attendre cet après-midi) **modérément exponentiels**
- Algorithmes paramétrés (pour cela il faut attendre à peu près une heure)

On verra quelques techniques majeures de l'algorithmique modérément exponentielle

Rappels rapides et notations (en vrac)

$O^*(\cdot)$: ignore les termes polynomiaux dans les expressions de complexité
Dans un graphe $G(V,E)$: $n = |V|$ et $m = |E|$
Ressources principales dans les algorithmes exacts :

- temps
- espace

La récurrence $T(n) = \sum_{i \leqslant p} T(n - c_i)$ vérifie $T(n) = O^*(c^n)$ où c est la plus grande solution positive de l'équation :

$$\sum_{i \leqslant p} x^{-c_i} = 1$$

L'idée de base et le principe de BELLMAN

C'est une des plus vieilles techniques pour produire des algorithmes exacts plus efficaces que l'énumération exhaustive

L'idée de base et le principe de BELLMAN

C'est une des plus vieilles techniques pour produire des algorithmes exacts plus efficaces que l'énumération exhaustive

Décomposer le problème en des sous-problèmes plus petits, garder en mémoire les solutions optimales de tous ces sous-problèmes et ...

L'idée de base et le principe de BELLMAN

C'est une des plus vieilles techniques pour produire des algorithmes exacts plus efficaces que l'énumération exhaustive

Décomposer le problème en des sous-problèmes plus petits, garder en mémoire les solutions optimales de tous ces sous-problèmes et ...

BELLMAN, 1949

Composer une solution optimale du problème en combinant les solutions (optimales) de ses sous-problèmes

TSP (1)

- Chercher un chemin hamiltonien de poids minimum entre le sommet 1 (arbitrairement fixé) et tout $i \neq 1$ et rajouter l'arête $(i, 1)$ afin d'obtenir un cycle
- Retenir le plus « léger » parmi les $n-1$ cycles ainsi produits

$\forall U \subset V, \forall i \in U \ (i \neq 1)$
$P(U, i)$: le chemin hamiltonien le plus « léger » entre 1 et i dans U

TSP (2)

La récurrence (HELD & KARP, 1962)

$\exists j \in U$ tel que j est l'avant dernier sommet dans $P(U, i)$
Par conséquent, calculer $P(U, i)$ revient à calculer $P(U \setminus \{i\}, j)$, $\forall j \in U \setminus \{i\}$, *i.e.* :

$$P(U, i) = \min_{\substack{j \in U \\ j \neq 1, i}} \{P(U \setminus \{i\}, j) + w(j, i)\}$$

TSP (2)

La récurrence (HELD & KARP, 1962)

$\exists j \in U$ tel que j est l'avant dernier sommet dans $P(U, i)$
Par conséquent, calculer $P(U, i)$ revient à calculer $P(U \setminus \{i\}, j)$, $\forall j \in U \setminus \{i\}$, *i.e.* :

$$P(U, i) = \min_{\substack{j \in U \\ j \neq 1, i}} \{P(U \setminus \{i\}, j) + w(j, i)\}$$

Calculer $P(U, i)$ revient à examiner tous les possibles $U \subseteq V$

TSP (2)

La récurrence (HELD & KARP, 1962)

$\exists j \in U$ tel que j est l'avant dernier sommet dans $P(U, i)$
Par conséquent, calculer $P(U, i)$ revient à calculer $P(U \setminus \{i\}, j)$, $\forall j \in U \setminus \{i\}$, *i.e.* :

$$P(U, i) = \min_{\substack{j \in U \\ j \neq 1, i}} \{P(U \setminus \{i\}, j) + w(j, i)\}$$

Calculer $P(U, i)$ revient à examiner tous les possibles $U \subseteq V$

On a donc une complexité $O^*(2^n)$

Le principe d'inclusion-exclusion

Une méthode très récente basée non pas sur la structure des instances mais plutôt sur une analyse combinatoire des propriétés de la solution

Idée de base

Reformuler la solution d'un problème comme expression d'une fonction proprement algébrique et tenter de calculer celle-ci rapidement

Elle a été appliqué surtout à des problèmes de graphes

COLORATION

COLORATION

Soit V, I deux ensembles et $(V_i)_{i \in I}$ une famille de sous-ensembles de V :

$$\left| \bigcup_{i \in I} V_i \right| = \sum_{i \in I} |V_i| - \sum_{i \neq j \in I} |V_i \cap V_j| + \ldots + (-1)^{|I|+1} \left| \bigcap_{i \in I} V_i \right|$$

C'est la formule de l'union d'ensembles

Par passage au complémentaire :

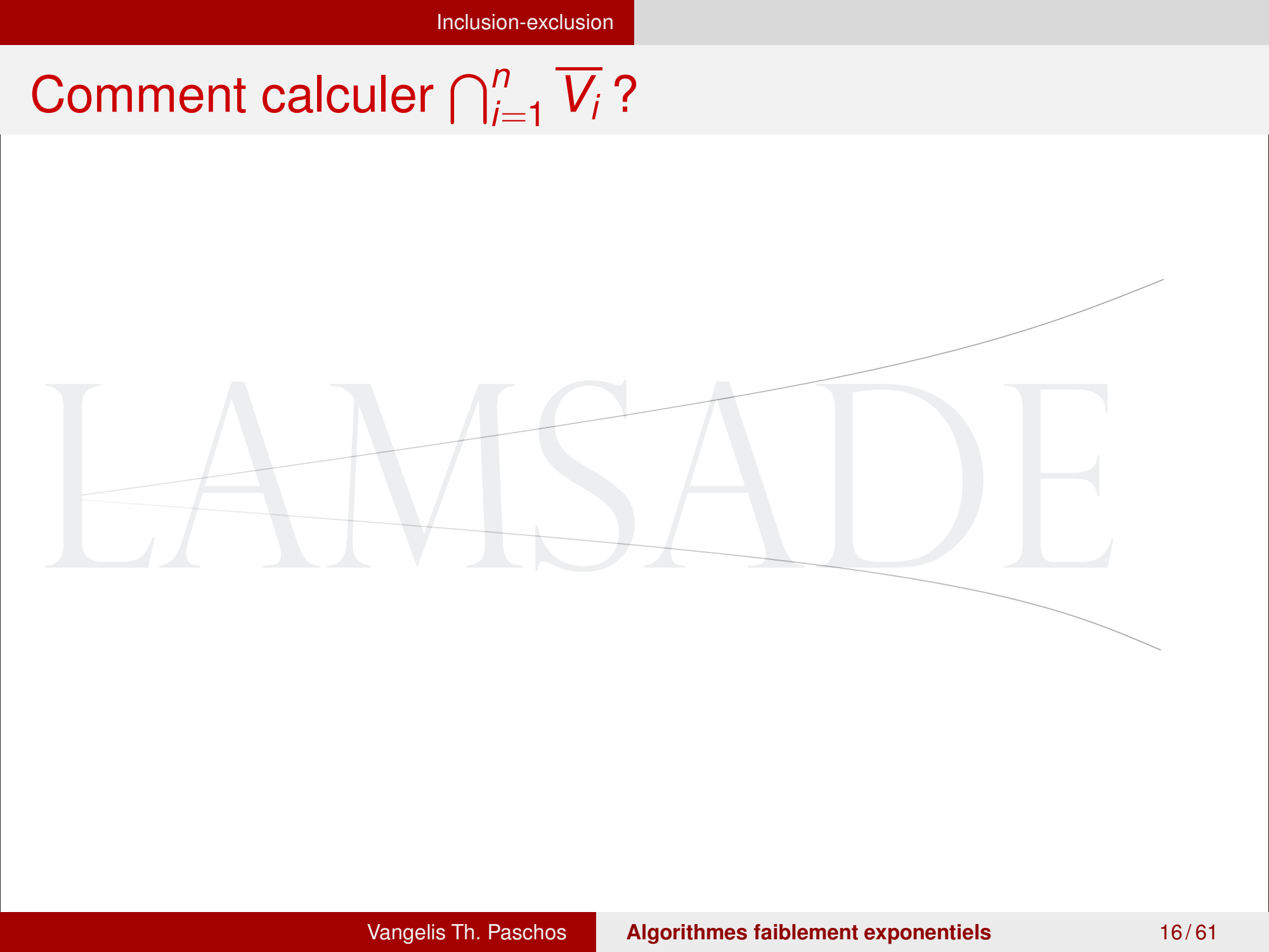
$$\left| \bigcap_{i \in I} (V \setminus V_i) \right| = \left| \bigcap_{i \in I} \overline{V_i} \right| = \sum_{X \subset I} (-1)^{|X|} \left| \bigcap_{i \in X} V_i \right|$$

Appliquons-la à la COLORATION

V : l'ensemble des sommets du graphe G (instance de COLORATION)
$\mathcal{F}$: la famille des stables de G
$V_i \subset \mathcal{F}^k$: l'ensemble des familles de k stables dont la réunion ne couvre pas le sommet i (on permet à une famille de compter plusieurs fois le même stable)
Alors :

- $\overline{V_i}$: l'ensemble des k-unions de stables qui couvrent i
- $\bigcap_{i=1}^{n} \overline{V_i}$: l'ensemble des k-unions de stables qui couvrent V
- $\bigcap_{i \in V} \overline{V_i} \neq \emptyset$, i.e., $\left|\bigcap_{i \in V} \overline{V_i}\right| > 0 \iff$ il existe au moins une famille de k stables qui couvre V, *une k-coloration*

Comment calculer $\bigcap_{i=1}^{n} \overline{V_i}$?



Comment calculer $\bigcap_{i=1}^{n} \overline{V_i}$?

- Rappelons-nous que $\left|\bigcap_{i\in V} \overline{V_i}\right| = \sum_{X\subset V} (-1)^{|X|} \left|\bigcap_{i\in X} V_i\right|$
- Poser $f(X) = (-1)^{|X|} \left|\bigcap_{i\in X} V_i\right|$

Comment calculer $\bigcap_{i=1}^{n} \overline{V_i}$?

- Rappelons-nous que $\left|\bigcap_{i \in V} \overline{V_i}\right| = \sum_{X \subset V} (-1)^{|X|} \left|\bigcap_{i \in X} V_i\right|$
- Poser $f(X) = (-1)^{|X|} \left|\bigcap_{i \in X} V_i\right|$
- **COLORATION revient à calculer** $\sum_{X \subseteq V} f(X)$

Comment calculer $\bigcap_{i=1}^{n} \overline{V_i}$?

- Rappelons-nous que $\left|\bigcap_{i \in V} \overline{V_i}\right| = \sum_{X \subset V} (-1)^{|X|} \left|\bigcap_{i \in X} V_i\right|$
- Poser $f(X) = (-1)^{|X|} \left|\bigcap_{i \in X} V_i\right|$
- **COLORATION revient à calculer** $\sum_{X \subseteq V} f(X)$
- Une famille des stables appartient à $V_i \iff i$ n'appartient à aucun de ses stables
- Par conséquent, elle appartient à $\bigcap_{i \in X} V_i \iff$ chacun de ses stables est totalement disjoint de X

Suite du calcul

Cette propriété est vérifiable indépendamment pour chaque stable :

$$\begin{aligned}\left|\bigcap_{i\in X} V_i\right| &= |S \in \mathcal{F}, S \cap X = \emptyset|^k \\ f(X) &= (-1)^{|X|}\left|\bigcap_{i\in X} V_i\right| \\ &= (-1)^{|X|}\,|S \in \mathcal{F}, S \cap X = \emptyset|^k\end{aligned}$$

COLORATION et espace polynomial

COLORATION et espace polynomial

- Calculer $f(X)$ indépendamment $\forall X$

COLORATION et espace polynomial

- Calculer $f(X)$ indépendamment $\forall X$
- $|S \in \mathcal{F}, S \cap X = \emptyset|$ est le nombre de stables dans le graphe $G[V \setminus X]$

COLORATION et espace polynomial

- Calculer $f(X)$ indépendamment $\forall X$
- $|S \in \mathcal{F}, S \cap X = \emptyset|$ est le nombre de stables dans le graphe $G[V \setminus X]$
- Il peut être calculé (ainsi que $f(X)$) en $O^*(2^{|V \setminus X|})$ (et **espace polynomial**)

COLORATION et espace polynomial

- Calculer $f(X)$ indépendamment $\forall X$
- $|S \in \mathcal{F}, S \cap X = \emptyset|$ est le nombre de stables dans le graphe $G[V \setminus X]$
- Il peut être calculé (ainsi que $f(X)$) en $O^*(2^{|V \setminus X|})$ (et **espace polynomial**)

$$\sum_{X \subseteq V} f(X) = O^* \left(\sum_{X \subseteq V} 2^{|V \setminus X|} \right) = O^* \left(\sum_{i=0}^{n} \binom{n}{i} 2^{n-i} \right) = O^*(3^n)$$

Il peut être amélioré

COLORATION et espace exponentiel

COLORATION et espace exponentiel

Idée

Calculer simultanément $f(X)$ par programmation dynamique

COLORATION et espace exponentiel

Idée

Calculer simultanément $f(X)$ par programmation dynamique

Soit $Y \subseteq V$ et $i \in Y$
Un stable $S \in Y$ soit contient i (et pas ses voisins $\Gamma(i)$), soit ne contient pas i :

$$|S \in \mathcal{F}, S \subseteq Y| = |S \in \mathcal{F}, S \subseteq Y \setminus \{\Gamma[i]\}| + |S \in \mathcal{F}, S \subseteq Y \setminus \{i\}|$$

COLORATION et espace exponentiel

Idée

Calculer simultanément $f(X)$ par programmation dynamique

Soit $Y \subseteq V$ et $i \in Y$
Un stable $S \in Y$ soit contient i (et pas ses voisins $\Gamma(i)$), soit ne contient pas i :

$$|S \in \mathcal{F}, S \subseteq Y| = |S \in \mathcal{F}, S \subseteq Y \setminus \{\Gamma[i]\}| + |S \in \mathcal{F}, S \subseteq Y \setminus \{i\}|$$

- Cette formule calcule $|S \in \mathcal{F}, S \subseteq Y|$ (considérant |Y| en ordre croissant) en temps et espace $O^*(2^n)$
- Poser $X = V \setminus Y$ et calculer $f(X)$

Le principe

Réduire récursivement l'instance en des sous-instances (plus petites) jusqu'à ce que l'instance survivante soit résoluble en temps polynomial

Le principe

Réduire récursivement l'instance en des sous-instances (plus petites) jusqu'à ce que l'instance survivante soit résoluble en temps polynomial

- Espace polynomial
- Peu de branches $\Longrightarrow$ complexité réduite
- Mais . . . le même sous-problème peut être traité plusieurs fois

STABLE

1. Etant donné un sommet i, soit il fait partie d'un stable maximum et $\Gamma(i)$ n'en fait pas partie, soit i lui-même n'en fait pas partie
2. Dans un graphe de degré maximum $\Delta \leqslant 2$ (collection des cycles et des chemins) STABLE est résoluble en temps polynomial

STABLE

1. Etant donné un sommet i, soit il fait partie d'un stable maximum et $\Gamma(i)$ n'en fait pas partie, soit i lui-même n'en fait pas partie
2. Dans un graphe de degré maximum $\Delta \leqslant 2$ (collection des cycles et des chemins) STABLE est résoluble en temps polynomial

$1 :\Longleftrightarrow T(n) \leqslant T(n-(\Delta+1)) + T(n-1)$

$2 :\Longleftrightarrow \Delta \geqslant 3$

STABLE

1. Etant donné un sommet i, soit il fait partie d'un stable maximum et $\Gamma(i)$ n'en fait pas partie, soit i lui-même n'en fait pas partie
2. Dans un graphe de degré maximum $\Delta \leqslant 2$ (collection des cycles et des chemins) STABLE est résoluble en temps polynomial

$1 :\Longleftrightarrow T(n) \leqslant T(n-(\Delta+1)) + T(n-1)$
$2 :\Longleftrightarrow \Delta \geqslant 3$

$$T(n) \leqslant T(n-1) + T(n-4) \Longleftrightarrow T(n) = O^*\left(1,385^n\right)$$

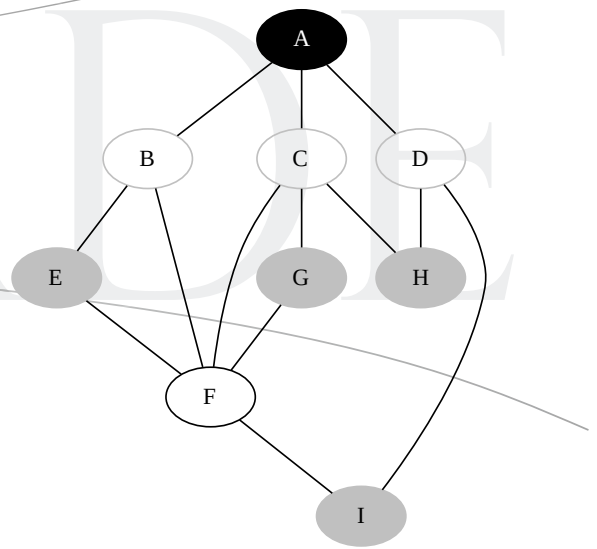
Règles de réduction

Configurations particulières où on peut réduire davantage l'instance en temps polynomial

Règles de réduction

Configurations particulières où on peut réduire davantage l'instance en temps polynomial

- En prenant A, alors B, C et D sont retirés
- On ajoute H (isolé) et E (de degré 1)
- F est retiré, G et I rajoutés



Si on a cela pour chaque branchement : $T(n) \leqslant T(n-9) + T(n-1)$

Troquer de l'espace pour du temps

Troquer de l'espace pour du temps

La solution d'un même sous-problème peut être recalculée plusieurs fois

Troquer de l'espace pour du temps

La solution d'un même sous-problème peut être recalculée plusieurs fois

Le remède : mixer branchement et programmation dynamique

Troquer de l'espace pour du temps

La solution d'un même sous-problème peut être recalculée plusieurs fois

Le remède : mixer branchement et programmation dynamique

- Résoudre tous les sous-problèmes d'une certaine taille k et stocker leurs solutions (dans un tableau) (temps $T(n,k)$)
- Brancher jusqu'à ce que la taille du problème devienne inférieure ou égale à k (temps $T(n-k)$)
- Compléter par la solution optimale du sous-problème stockée dans le tableau

Troquer de l'espace pour du temps

La solution d'un même sous-problème peut être recalculée plusieurs fois

Le remède : mixer branchement et programmation dynamique

- Résoudre tous les sous-problèmes d'une certaine taille k et stocker leurs solutions (dans un tableau) (temps $T(n,k)$)
- Brancher jusqu'à ce que la taille du problème devienne inférieure ou égale à k (temps $T(n-k)$)
- Compléter par la solution optimale du sous-problème stockée dans le tableau

Temps $T(n) \leqslant T(n,k) + T(n-k)$

MIN STABLE MAXIMAL

MIN STABLE MAXIMAL

Chaque sommet v sera dominé soit par lui-même, soit par un de ses voisins :

$$\text{opt}(G) = \min_{u_i \in \Gamma[v]} \{\text{opt}\left(G\left[V \setminus \Gamma\left[u_i\right]\right]\right)\}$$

Ce qui donne, en branchant sur un sommet de degré minimum δ :

$$T(n) \leqslant (\delta+1)T(n-(\delta+1)) \overset{\delta \geqslant 2}{\Longrightarrow} T(n) = O^*\left(3^{n/3}\right)$$

En mémoïsant avec $k = n/10$:

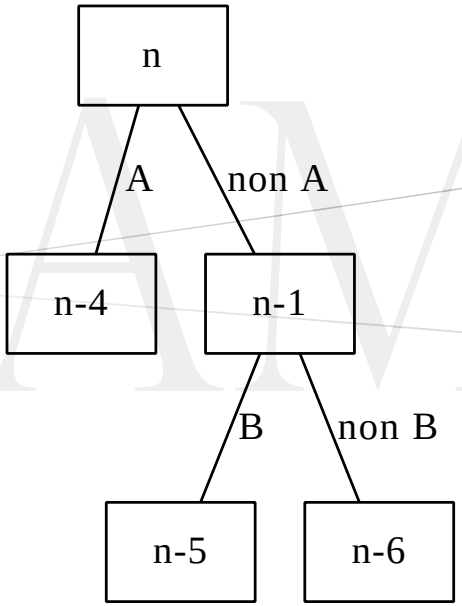
$T(n,k) = O^*\left(\binom{n}{n/10}\right) = O^*\left(1,39^n\right)$

$T(n-k) = T(9n/10) = O^*\left(3^{3n/10}\right) = O^*\left(1,40^n\right)$

Reculer pour mieux sauter

- Plusieurs fois, lorsqu'un certain « mauvais » branchement est effectué, l'instance qui en résulte possède certaines propriétés qui assurent une série de meilleurs branchements dans la phase suivante
- Cela améliore le temps de calcul mais, très souvent, au prix d'un surcoût dans l'analyse (qui peut devenir une analyse fatidique des cas et peu élégante)

Un schéma de branchement multiple



Quand A est éliminé, l'instance est peu réduite
Mais le nouveau graphe (branche de droite) possède une propriété qui autorise un second meilleur branchement :
$T(n-4) + T(n-5) + T(n-6)$

STABLE dans les graphes avec $\Delta = 3$

STABLE dans les graphes avec $\Delta = 3$

Pliage d'un sommet (*Vertex folding*)

STABLE dans les graphes avec $\Delta = 3$

Pliage d'un sommet (*Vertex folding*)

- Soit $u : \Gamma(u) = \{x, y\}$ et x, y non-adjacents

STABLE dans les graphes avec $\Delta = 3$

Pliage d'un sommet (*Vertex folding*)

- Soit $u : \Gamma(u) = \{x, y\}$ et x, y non-adjacents
- Dans un stable maximum on peut supposer que soit on prend u, soit on prend x et y

STABLE dans les graphes avec $\Delta = 3$

Pliage d'un sommet (*Vertex folding*)

- Soit $u : \Gamma(u) = \{x, y\}$ et x, y non-adjacents
- Dans un stable maximum on peut supposer que soit on prend u, soit on prend x et y
- On peut plier u, *i.e.*, l'enlever et contracter x et y en un seul sommet w avec $\Gamma(w) = \Gamma(x) \cup \Gamma(y)$

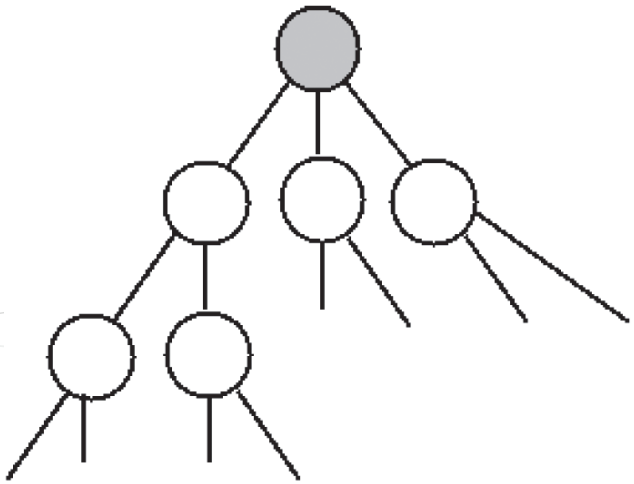
STABLE dans les graphes avec $\Delta = 3$

Pliage d'un sommet (*Vertex folding*)

- Soit $u : \Gamma(u) = \{x, y\}$ et x, y non-adjacents
- Dans un stable maximum on peut supposer que soit on prend u, soit on prend x et y
- On peut plier u, *i.e.*, l'enlever et contracter x et y en un seul sommet w avec $\Gamma(w) = \Gamma(x) \cup \Gamma(y)$

Par conséquent, même dans un graphe avec $\Delta = 3$ on peut créer des « sommets » de degré $\geqslant 4$

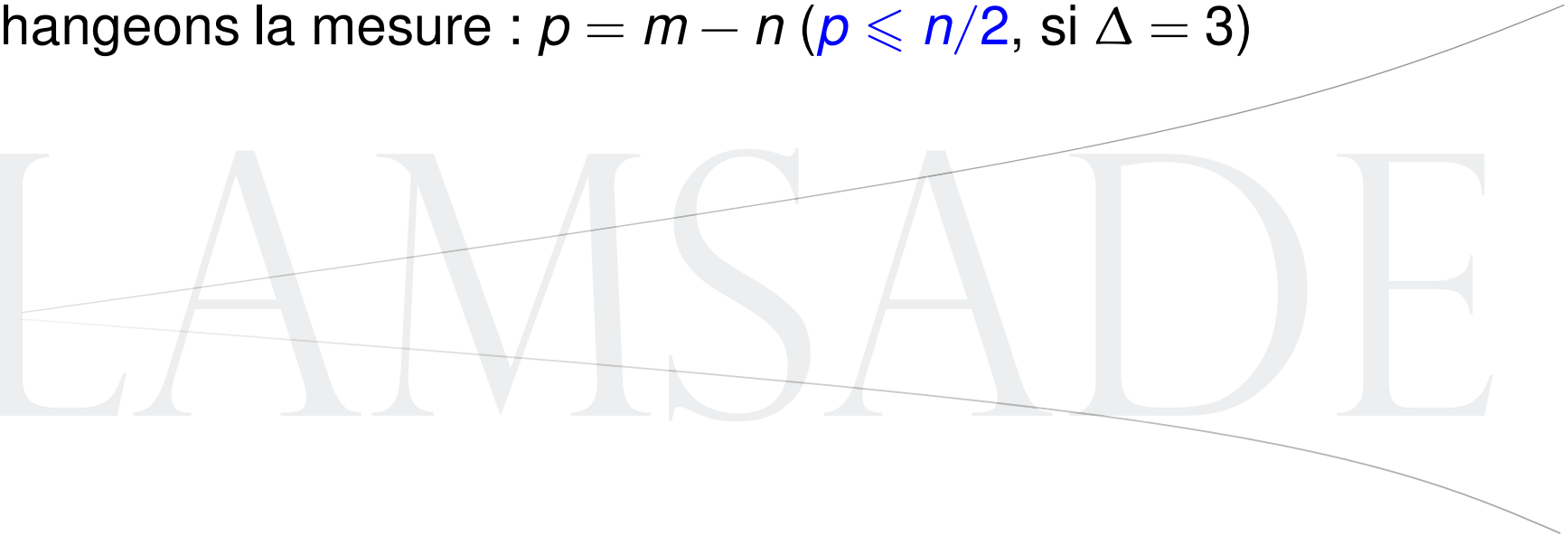
Un exemple de pliage



Après avoir supprimé le sommet gris, nous pouvons plier le sommet le plus à gauche pour créer un « **super-sommet** » de degré 4

Brancher sur un sommet de degré au moins 4

Changeons la mesure : $p = m - n$ ($p \leqslant n/2$, si $\Delta = 3$)



Brancher sur un sommet de degré au moins 4

Changeons la mesure : $p = m - n$ ($p \leqslant n/2$, si $\Delta = 3$)

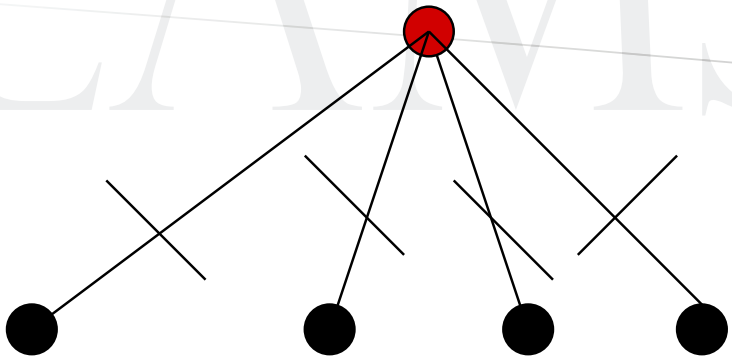
Branchons sur un sommet de degré maximum (soit $\geqslant 4$) :

Brancher sur un sommet de degré au moins 4

Changeons la mesure : $p = m - n$ ($p \leqslant n/2$, si $\Delta = 3$)

Branchons sur un sommet de degré maximum (soit $\geqslant 4$) :

Si on ne le prend pas :

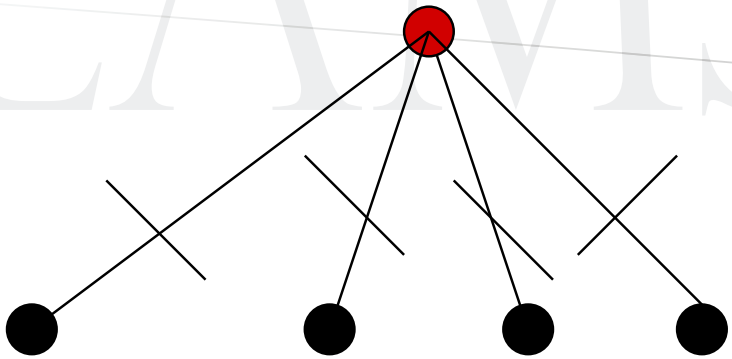


Brancher sur un sommet de degré au moins 4

Changeons la mesure : $p = m - n$ ($p \leqslant n/2$, si $\Delta = 3$)

Branchons sur un sommet de degré maximum (soit $\geqslant 4$) :

Si on ne le prend pas :



- $n \leftarrow n - 1$
- $m \leftarrow m - 4$
- $p \leftarrow p - 3$

Brancher sur un sommet de degré au moins 4 (suite)

Si on le prend, en appliquant une règle de réduction on aura au pire des cas (croyez moi sur parole) :

- $n \leftarrow n - 5$
- $m \leftarrow m - 10$
- $p \leftarrow p - 5$

$$T(p) \leqslant T(p-3) + T(p-5) \Longrightarrow T(p) \leqslant O^*(1,194^p)$$

Brancher sur un sommet de degré au moins 4 (suite)

Si on le prend, en appliquant une règle de réduction on aura au pire des cas (croyez moi sur parole) :

- $n \leftarrow n-5$
- $m \leftarrow m-10$
- $p \leftarrow p-5$

$T(p) \leqslant T(p-3)+T(p-5) \Longrightarrow T(p) \leqslant O^*(1,194^p)$

Si il y a un triangle : $T(p) \leqslant 2T(p-4) \Longrightarrow T(p) \leqslant O^*(2^{p/4})$ (encore une fois . . . sur parole)

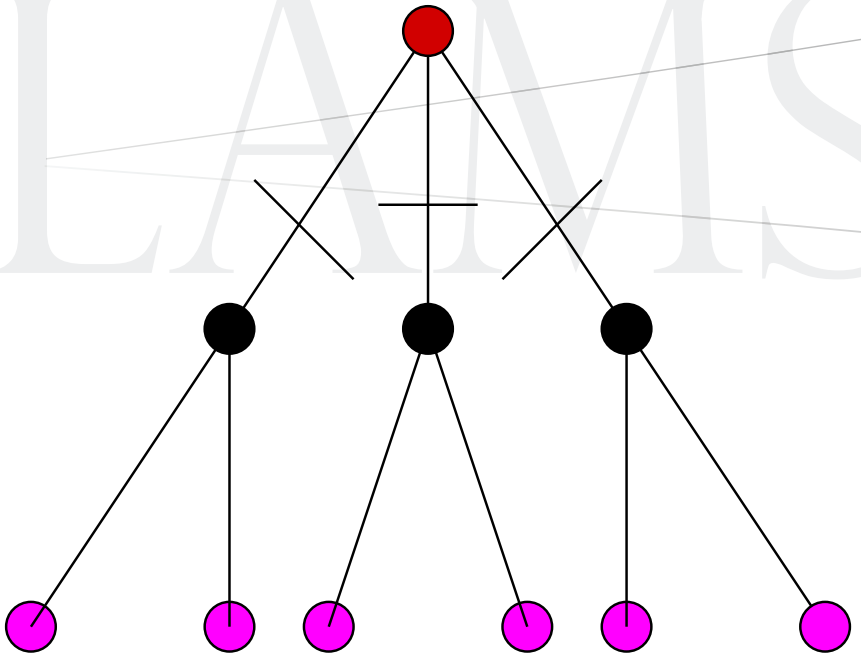
Graphes cubiques (1)

Le pire cas est quand le graphe est cubique et sans triangle

Graphes cubiques (1)

Le pire cas est quand le graphe est cubique et sans triangle

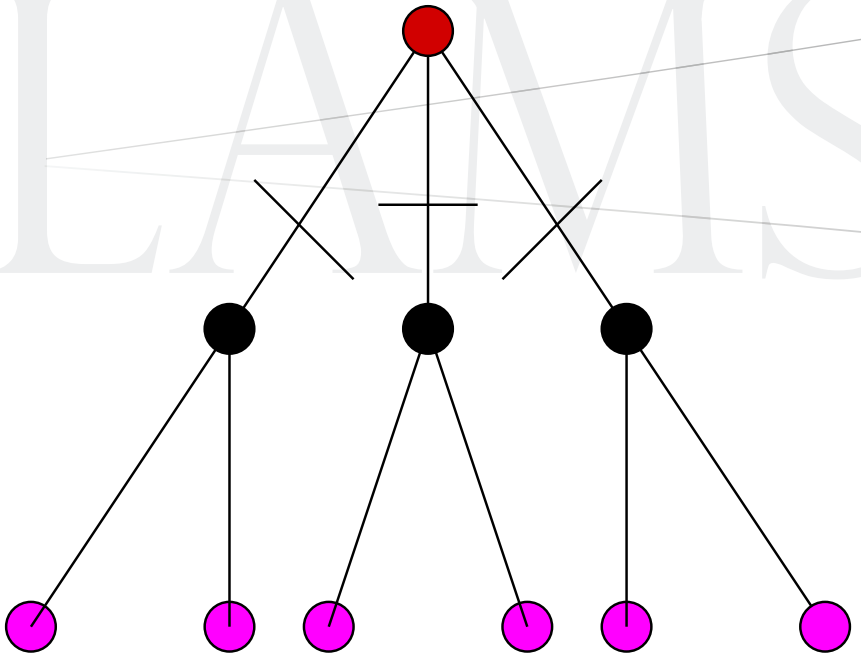
Si on ne prend pas le sommet rouge :



Graphes cubiques (1)

Le pire cas est quand le graphe est cubique et sans triangle

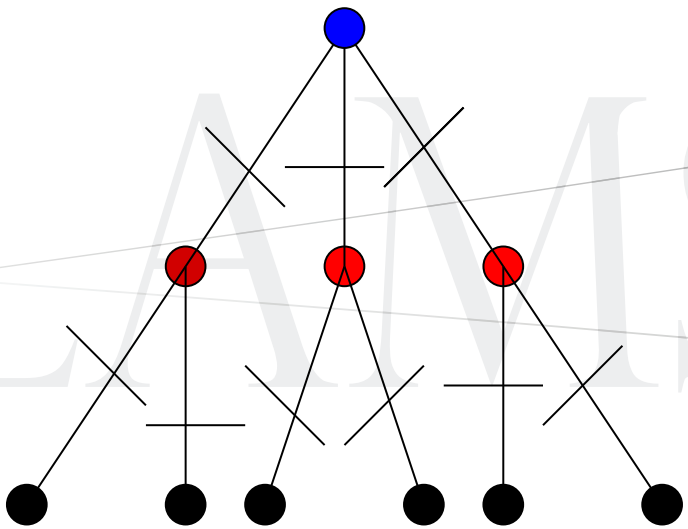
Si on ne prend pas le sommet rouge :



$$n \leftarrow n - 1$$
$$m \leftarrow m - 3$$
$$p \leftarrow p - 2$$

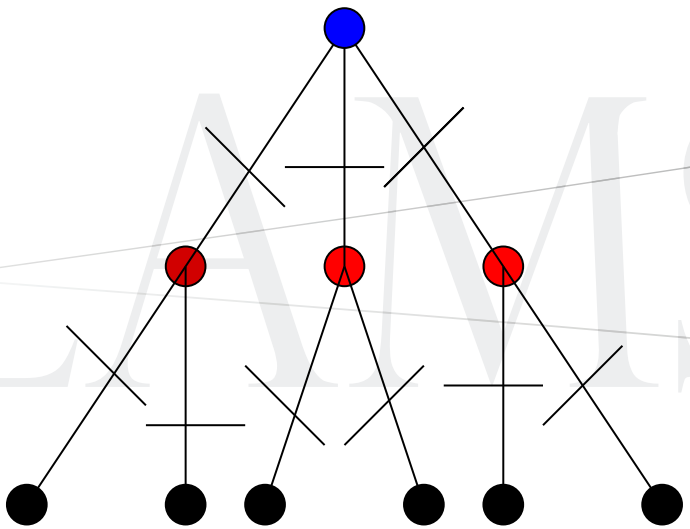
Graphes cubiques (2)

Si on le prend :



Graphes cubiques (2)

Si on le prend :



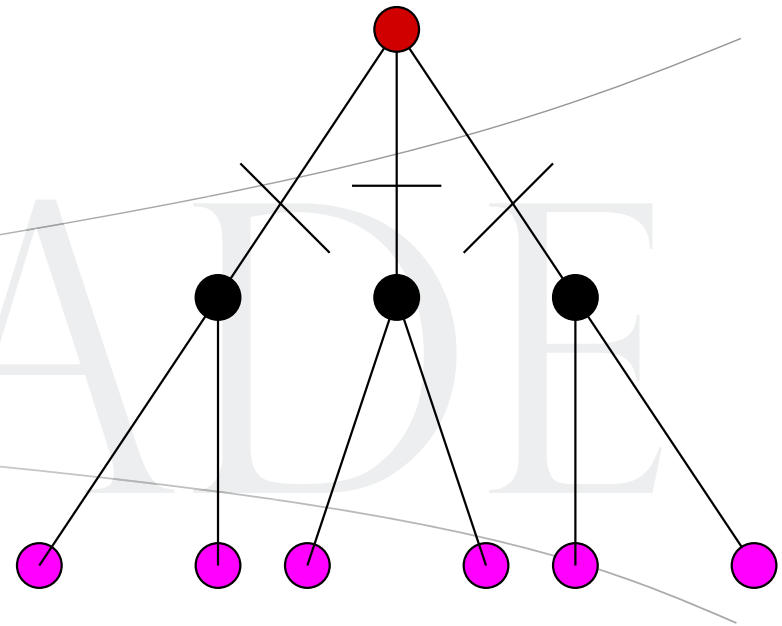
$$n \leftarrow n-4$$
$$m \leftarrow m-9$$
$$p \leftarrow p-5$$

$$T(p) \leqslant T(p-2) + T(p-5)$$

Graphes cubiques - branchement multiple

Mais . . .

Après rejet du sommet rouge les sommets noirs sont de degré 2
On les plie et nous avons au moins un sommet de degré $\geqslant 4$



$$\begin{aligned} T(p) &\leqslant T(p-5) + T((p-2)-3) + T((p-2)-5) \\ &= 2T(p-5) + T(p-7) \end{aligned}$$

$$T(p) \leqslant O^*(1,2175^p) = O^*(1,1034^n)$$

Les limites de la méthode classique (brute) du branchement

Les limites de la méthode classique (brute) du branchement

- Chaque amélioration successive augmente considérablement la complication de l'analyse

Les limites de la méthode classique (brute) du branchement

- Chaque amélioration successive augmente considérablement la complication de l'analyse
- Dans l'exemple précédent on a appliqué règles de réduction, on a plié des sommets, on appliqué des branchements multiples, on a étudié des cas (triangles ou pas, seuils des degrés, ...), ...

Et le problème ne concernait que les graphes à $\Delta = 3$! ! !

Les limites de la méthode classique (brute) du branchement

- Chaque amélioration successive augmente considérablement la complication de l'analyse
- Dans l'exemple précédent on a appliqué règles de réduction, on a plié des sommets, on appliqué des branchements multiples, on a étudié des cas (triangles ou pas, seuils des degrés, ...), ...

Et le problème ne concernait que les graphes à $\Delta = 3$! ! !

Pouvons nous avoir un outil qui évite les longues analyses des cas pathologiques ?

L'idée de base et les ingrédients

L'idée de base

Equilibrer la complexité au pire des cas entre branchements favorables et défavorables

L'idée de base et les ingrédients

L'idée de base

Equilibrer la complexité au pire des cas entre branchements favorables et défavorables

- Des mesures de progression plus informatives sur la structure de l'instance que sa taille seule (densité d'un graphe, # sommets d'un certain degré, ...)

L'idée de base et les ingrédients

L'idée de base

Equilibrer la complexité au pire des cas entre branchements favorables et défavorables

- Des mesures de progression plus informatives sur la structure de l'instance que sa taille seule (densité d'un graphe, # sommets d'un certain degré, ...)
- Partitionnement des instances du problème suivant une propriété structurelle

L'idée de base et les ingrédients

L'idée de base

Equilibrer la complexité au pire des cas entre branchements favorables et défavorables

- Des mesures de progression plus informatives sur la structure de l'instance que sa taille seule (densité d'un graphe, # sommets d'un certain degré, ...)
- Partitionnement des instances du problème suivant une propriété structurelle
- Différenciation de la contribution d'un élément suivant quelques caractéristiques structurelles propres

Une application simple sur STABLE (encore)

Une application simple sur STABLE (encore)

- Soit le branchement simple : prendre un sommet de degré maximum et l'enlever du graphe ainsi que ses voisins, ou ne pas le prendre et l'enlever du graphe

Une application simple sur STABLE (encore)

- Soit le branchement simple : prendre un sommet de degré maximum et l'enlever du graphe ainsi que ses voisins, ou ne pas le prendre et l'enlever du graphe
- Partitionner les graphes en deux sous-familles : celle, $\mathcal{G}_1$, des graphes à $\Delta \geqslant 4$ et celle, $\mathcal{G}_2$, des graphes à $\Delta \leqslant 3$

Une application simple sur STABLE (encore)

- Soit le branchement simple : prendre un sommet de degré maximum et l'enlever du graphe ainsi que ses voisins, ou ne pas le prendre et l'enlever du graphe
- Partitionner les graphes en deux sous-familles : celle, $\mathcal{G}_1$, des graphes à $\Delta \geqslant 4$ et celle, $\mathcal{G}_2$, des graphes à $\Delta \leqslant 3$
- Pour les branchements nous avons :
  - $\mathcal{G}_1 \longrightarrow T(n) \leqslant T(n-1) + T(n-5)$
  - $\mathcal{G}_2 \longrightarrow T(n) \leqslant T(n-1) + T(n-4)$

  Le pire est le deuxième avec $O^*(1,381^n)$

Changer la mesure

Changer la mesure

- Donner à tous les sommets de degré 2 (soit n_2 leur nombre) poids $1 - \epsilon$, $\epsilon < 0,5$

Changer la mesure

- Donner à tous les sommets de degré 2 (soit n_2 leur nombre) poids $1 - \epsilon$, $\epsilon < 0,5$
- Donner à tous les sommets de degré $\geqslant 3$ (soit $n_{\geqslant 3}$ leur nombre) poids 1

Changer la mesure

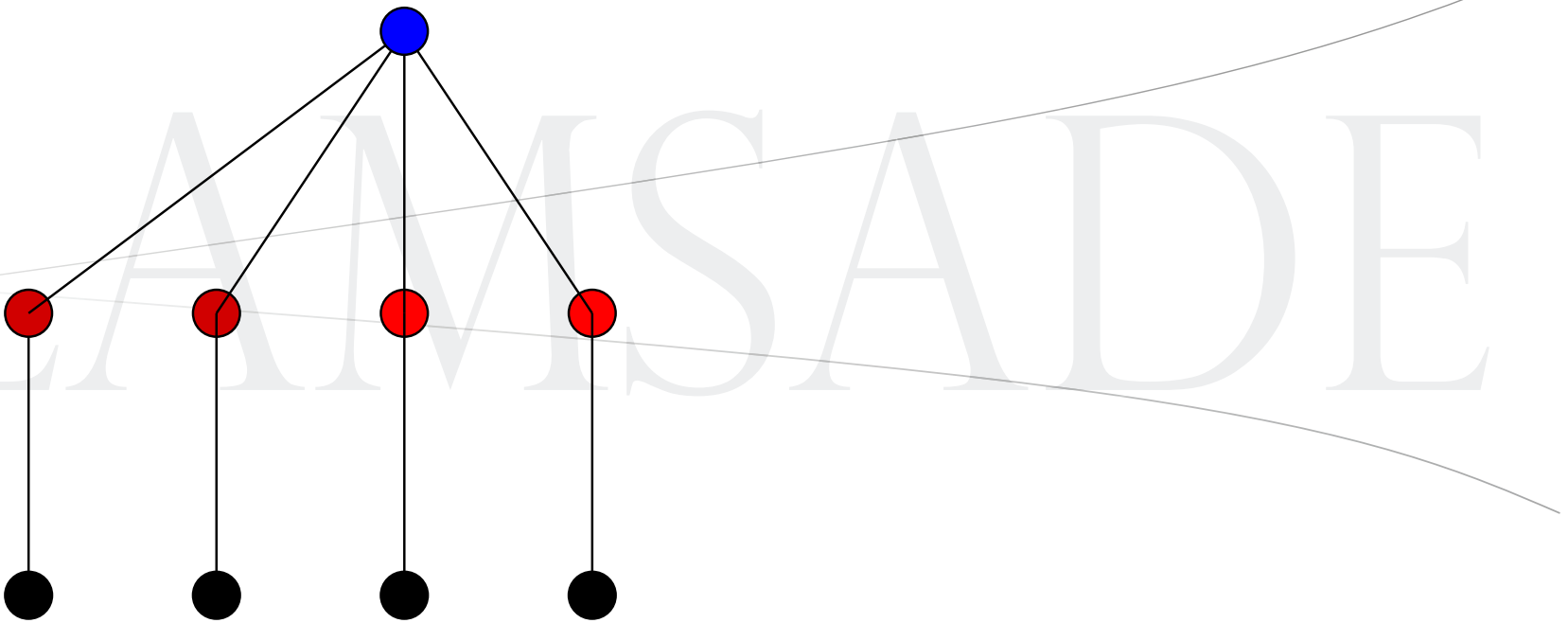
- Donner à tous les sommets de degré 2 (soit n_2 leur nombre) poids $1 - \epsilon$, $\epsilon < 0,5$
- Donner à tous les sommets de degré $\geqslant 3$ (soit $n_{\geqslant 3}$ leur nombre) poids 1
- Les sommets de degré $\leqslant 1$ rentrent dans la solution sans branchement

Changer la mesure

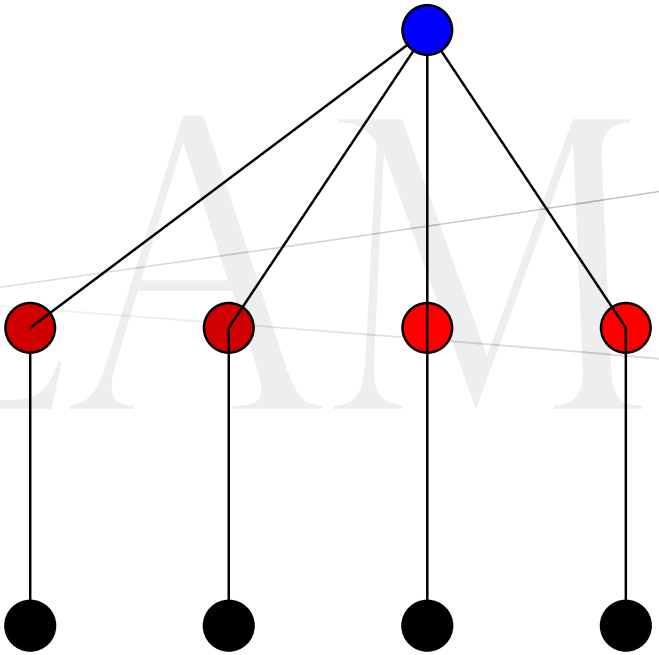
- Donner à tous les sommets de degré 2 (soit n_2 leur nombre) poids $1 - \epsilon$, $\epsilon < 0,5$
- Donner à tous les sommets de degré $\geqslant 3$ (soit $n_{\geqslant 3}$ leur nombre) poids 1
- Les sommets de degré $\leqslant 1$ rentrent dans la solution sans branchement
- Considérer la mesure $p = n_{\geqslant 3} + (1 - \epsilon)n_2 \leqslant n$

Branchement à $\mathcal{G}_1$

Branchement à $\mathcal{G}_1$



Branchement à $\mathcal{G}_1$



Ici (voisins du sommet bleu tous à degré 2)

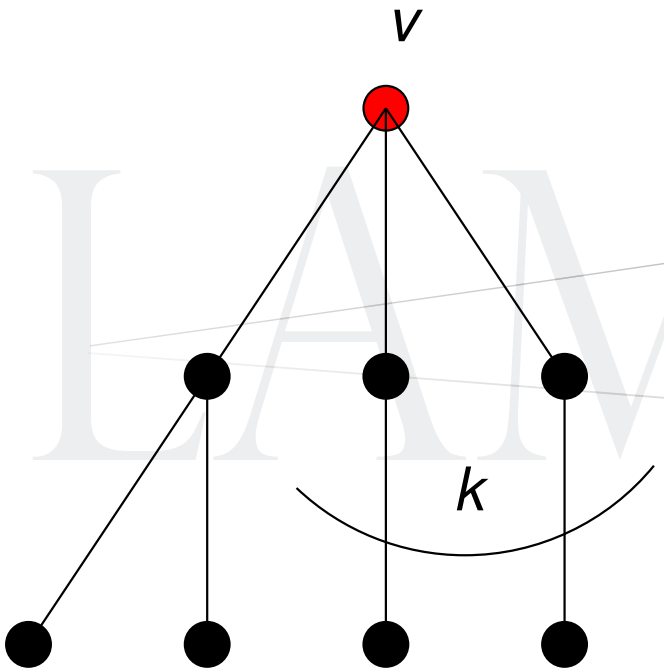
$p \longrightarrow p-1-4(1-\epsilon) = p-5+4\epsilon$

$T(p) \leqslant T(p-1) + T(p-5+4\epsilon)$

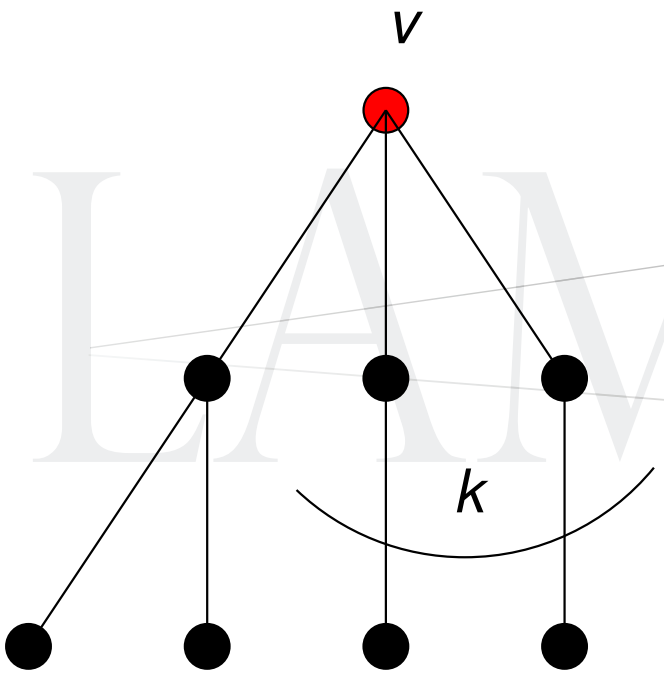
$(\geqslant T(p-1) + T(p-5))$

Branchement à $\mathcal{G}_2$ (1)

Branchement à $\mathcal{G}_2$ (1)



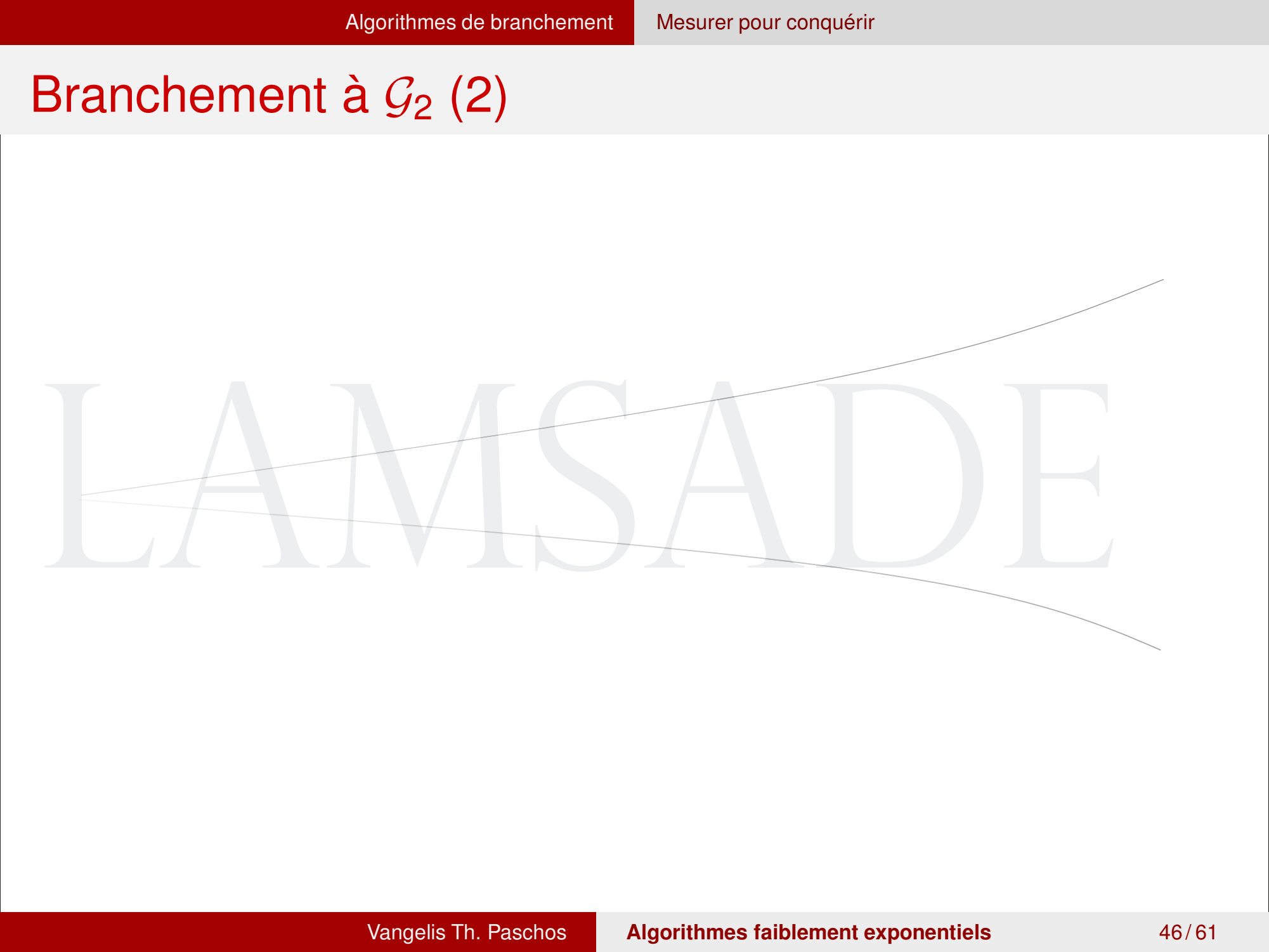
Branchement à $\mathcal{G}_2$ (1)



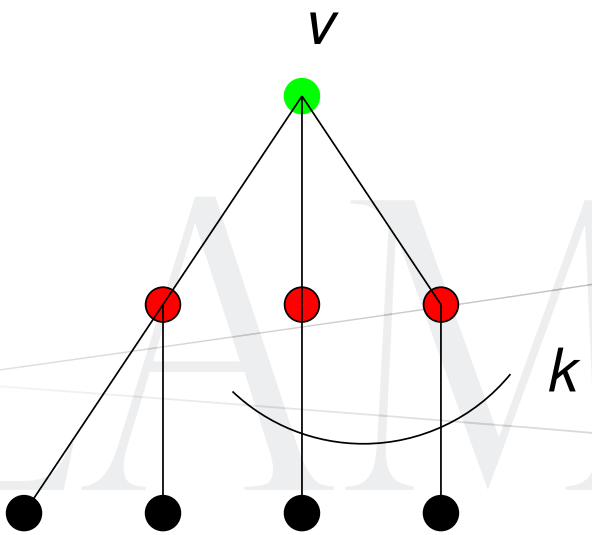
v est fixé
k sommets de degré 2 disparaissent devenant de degré 1
gain $k(1-\epsilon)$
$3-k$ sommets de degré 3 deviennent de degré 2
gain $(3-k)-(3-k)(1-\epsilon)=(3-k)\epsilon$

$$p \longleftarrow p-1-k(1-\epsilon)-(3-k)\epsilon$$

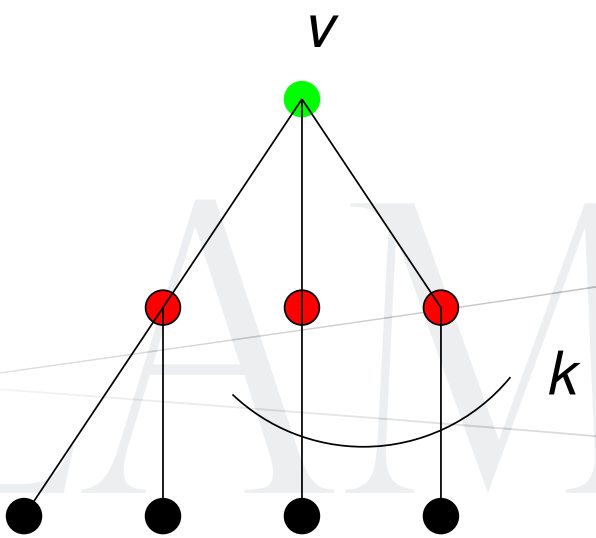
Branchement à $\mathcal{G}_2$ (2)



Branchement à $\mathcal{G}_2$ (2)

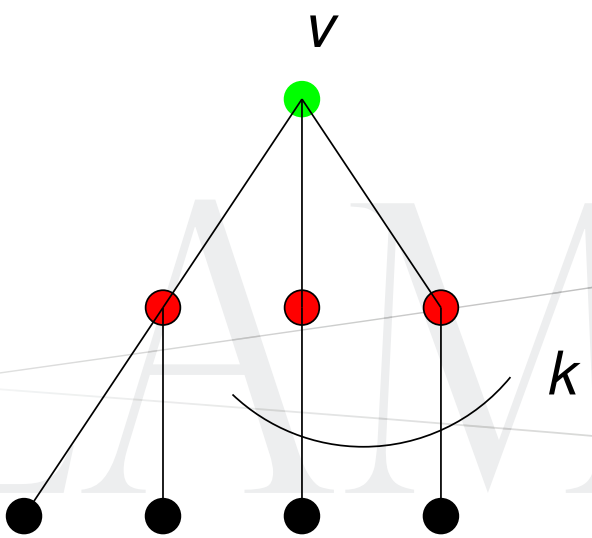


Branchement à $\mathcal{G}_2$ (2)



v est fixé
k sommets de degré 2 disparaissent
gain $k(1-\epsilon)$
$3-k$ sommets de degré 3 disparaissent
gain $(3-k)$
$p \longleftarrow p-4+k\epsilon$

Branchement à $\mathcal{G}_2$ (2)



v est fixé
k sommets de degré 2 disparaissent
gain $k(1-\epsilon)$
$3-k$ sommets de degré 3 disparaissent
gain $(3-k)$
$p \longleftarrow p-4+k\epsilon$

$$T(p) \leqslant \max_{1 \leqslant k \leqslant 3} \{T(p-1-k(1-\epsilon)-(3-k)\epsilon)+T(p-4+k\epsilon)\}$$

Pour $\epsilon = 0,085$, $T(p) = O^*(1,342^p) \leqslant O^*(1,342^n)$

M & C (avantages)

- Par le jeu des poids et le choix de la mesure, « diminuer » le déséquilibre entre les divers « pires » branchements (les bons d'entre eux deviennent moins bons et les mauvais moins mauvais)
- Si les mauvais branchements modifient l'instance de façon à ce qu'ils produisent des situations où des bons branchements deviennent possibles, cela se capitalise « automatiquement » sans devoir recourir aux analyses des cas fastidieuses, comme c'est le cas des branchements standards

Questions fondamentales

Questions fondamentales

Measure and conquer a-t-il donné des résultats intéressants ?

Les meilleurs résultats connus pour la COUVERTURE D'ENSEMBLES, l'ENCEMBLE DOMINANT, STABLE, MIN STABLE MAXIMAL, etc., sont basés sur le *measure and conquer*, ou ces améliorations

Questions fondamentales

Measure and conquer a-t-il donné des résultats intéressants ?

Les meilleurs résultats connus pour la COUVERTURE D'ENSEMBLES, l'ENCEMBLE DOMINANT, STABLE, MIN STABLE MAXIMAL, etc., sont basés sur le *measure and conquer*, ou ces améliorations

QUAND CE TYPE VA-T-IL FINIR ? ? ? ?

L'idée de base

Traitement préalable des données

Traiter préalablement les données (en les triant, ou en faisant des calculs rapides, ou en les partitionnant, ou . . .) de manière à ce que le problème soit résolu sur des instances plus petites et que les solutions partielles soient composés en une solution optimale plus rapidement

C'est une technique assez courue aussi en approximation polynomiale

SAC-A-DOS

$$I = \begin{cases} \max & \sum_{i=1}^{n} a_i x_i \\ & \sum_{i=1}^{n} c_i x_i \leqslant b \end{cases}$$

SAC-A-DOS

$$I = \begin{cases} \max & \sum_{i=1}^{n} a_i x_i \\ & \sum_{i=1}^{n} c_i x_i \leqslant b \end{cases}$$

Résoluble aussi par programmation dynamique en $O(n^2 a_{\max} \log c_{\max})$ et par énumération exhaustive en $O^*(2^n)$

SAC-A-DOS

$$I = \begin{cases} \max & \sum_{i=1}^{n} a_i x_i \\ & \sum_{i=1}^{n} c_i x_i \leqslant b \end{cases}$$

Résoluble aussi par programmation dynamique en $O(n^2 a_{\max} \log c_{\max})$ et par énumération exhaustive en $O^*(2^n)$

Soit I l'ensemble des objets
$\forall J \subseteq I$:
– $b(J)$ le « poids total » de J
– $a(J)$ la « valeur totale » de J

L'algorithme

L'algorithme

1. Partitionner les objets en deux sous-ensembles I_1 et I_2 ($I = I_1 \cup I_2$) de taille égale

L'algorithme

1. Partitionner les objets en deux sous-ensembles I_1 et I_2 ($I = I_1 \cup I_2$) de taille égale
2. $\forall J \subseteq I_k$, $k = 1, 2$, calculer $b(J)$ et $a(J)$

L'algorithme

1. Partitionner les objets en deux sous-ensembles I_1 et I_2 ($I = I_1 \cup I_2$) de taille égale
2. $\forall J \subseteq I_k$, $k = 1, 2$, calculer $b(J)$ et $a(J)$
3. $\forall J \subseteq I_k$, trier les $b(J)$ (en cas d'égalité donner la priorité au plus petit $a(J)$) et éliminer les sous-ensembles dominés

L'algorithme

1. Partitionner les objets en deux sous-ensembles I_1 et I_2 ($I = I_1 \cup I_2$) de taille égale
2. $\forall J \subseteq I_k$, $k = 1, 2$, calculer $b(J)$ et $a(J)$
3. $\forall J \subseteq I_k$, trier les $b(J)$ (en cas d'égalité donner la priorité au plus petit $a(J)$) et éliminer les sous-ensembles dominés
4. $\forall J_1 \in I_1$, trouver, par recherche dichotomique, $J_2 \in I_2$ qui maximise $a(J_1) + a(J_2)$ avec $b(J_1) + b(J_2) \leqslant b$ et retenir le couple (J_1, J_2) avec le plus grand a

La complexité

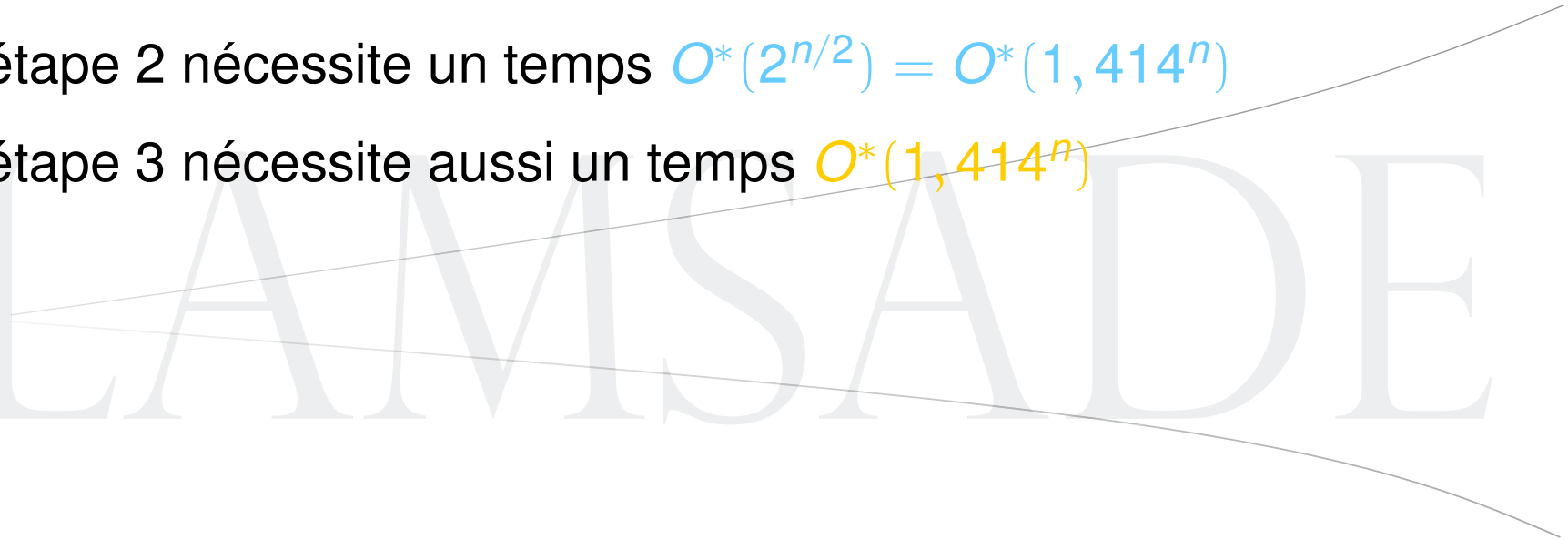
La complexité

L'étape 2 nécessite un temps $O^*(2^{n/2}) = O^*(1,414^n)$

La complexité

L'étape 2 nécessite un temps $O^*(2^{n/2}) = O^*(1,414^n)$

L'étape 3 nécessite aussi un temps $O^*(1,414^n)$



La complexité

L'étape 2 nécessite un temps $O^*(2^{n/2}) = O^*(1,414^n)$

L'étape 3 nécessite aussi un temps $O^*(1,414^n)$

A l'étape 4 la recherche dichotomique se fait en temps polynomial, donc le temps global de 4 est aussi $O^*(1,414^n)$

La complexité

L'étape 2 nécessite un temps $O^*(2^{n/2}) = O^*(1,414^n)$

L'étape 3 nécessite aussi un temps $O^*(1,414^n)$

A l'étape 4 la recherche dichotomique se fait en temps polynomial, donc le temps global de 4 est aussi $O^*(1,414^n)$

Donc

Complexité globale $O^*(1,414^n)$

L'idée de base

- Partir d'une solution réalisable qui se trouve à une certaine distance (défini de façon appropriée) de la solution « optimale »
- Produire une nouvelle solution en changeant seulement une petite partie de la solution précédente de façon à réduire la distance de cette nouvelle solution de la solution optimale
- Réitérer jusqu'à trouver la solution optimale

3-SAT (1)

3-SAT (1)

Pour une affectaion τ :
$\mathcal{H}(\tau, d)$ l'ensemble des τ' à distance de HAMMING $\leqslant d$ de τ

Etant donnés Φ une instance 3-SAT, τ et $d \in \mathbb{N}^+$, décider si $\mathcal{H}(\tau, d)$ contient un modèle pour Φ peut se faire en $O^*(3^d)$

3-SAT (1)

Pour une affectaion τ :
$\mathcal{H}(\tau, d)$ l'ensemble des τ' à distance de HAMMING $\leqslant d$ de τ

Etant donnés Φ une instance 3-SAT, τ et $d \in \mathbb{N}^+$, décider si $\mathcal{H}(\tau, d)$ contient un modèle pour Φ peut se faire en $O^*(3^d)$

En effet, sur une clause non satisfaite par τ, si on change la valeur d'une de ses variables, pour au moins un des changements la distance de HAMMING du résultat par rapport au modèle (s'il existe) diminuera $T(d) \leqslant 3T(d-1) \Longrightarrow T(d) = O^*(3^d)$

3-SAT (2)

Si 1^n et 0^n les affectations à toute coordonnée égale à 1 où à 0, toute affectation τ (*a fortiori* un modèle) appartiendra soit à $\mathcal{H}(1^n, n/2)$ soit à $\mathcal{H}(0^n, n/2)$

3-SAT (2)

Si 1^n et 0^n les affectations à toute coordonnée égale à 1 où à 0, toute affectation τ (*a fortiori* un modèle) appartiendra soit à $\mathcal{H}(1^n, n/2)$ soit à $\mathcal{H}(0^n, n/2)$

Et le décider peut se faire en $O^*(3^{n/2}) = O^*(1,73^n)$

J'ai presque fini (sérieusement !)

J'ai presque fini (sérieusement !)

- Plus d'outils d'analyse et (si possible) de techniques de conception d'algorithmes

J'ai presque fini (sérieusement !)

- Plus d'outils d'analyse et (si possible) de techniques de conception d'algorithmes
- Une structure pour la résolution modérément exponentielle

J'ai presque fini (sérieusement !)

- Plus d'outils d'analyse et (si possible) de techniques de conception d'algorithmes
- Une structure pour la résolution modérément exponentielle

Deux hypothèses de complexité (sous la conjecture **P** $\neq$ **NP**)

ETH et **SETH**

- **ETH** : $\forall k \geqslant 3$, $\exists s_k \in \mathbb{R}^+$, tel que k-SAT n'est pas résoluble en moins que $O^*(2^{s_k n})$
- **SETH** : $s_k \longrightarrow 1$, pour $k \longrightarrow \infty$

Dernier transparent

Dernier transparent

- Démontrer des bornes inférieures sous **SETH** pour des problèmes paradigmatiques (STABLE, COLORATION, ...)

Dernier transparent

- Démontrer des bornes inférieures sous **SETH** pour des problèmes paradigmatiques (STABLE, COLORATION, ...)
- Parmi les problèmes COLORATION, SAT, TSP, COUPE, etc., dont le meilleur algorithme connu est en $O^*(2^n)$, la résolution en $O^*(2^{cn})$, $c < 1$, pour l'un d'entre eux implique-t-elle la résolution en $O^*(2^{c'n})$, $c' < 1$ pour un (ou plusieurs) autre(s) ?

Dernier transparent

- Démontrer des bornes inférieures sous **SETH** pour des problèmes paradigmatiques (STABLE, COLORATION, ...)
- Parmi les problèmes COLORATION, SAT, TSP, COUPE, etc., dont le meilleur algorithme connu est en $O^*(2^n)$, la résolution en $O^*(2^{cn})$, $c < 1$, pour l'un d'entre eux implique-t-elle la résolution en $O^*(2^{c'n})$, $c' < 1$ pour un (ou plusieurs) autre(s) ?
- Peut on, en assumant l'insolvabilité des problèmes précédents à mieux que $O^*(2^n)$, produire des bornes inférieures de complexité pour d'autres problèmes ?

Dernier transparent

- Démontrer des bornes inférieures sous **SETH** pour des problèmes paradigmatiques (STABLE, COLORATION, ...)
- Parmi les problèmes COLORATION, SAT, TSP, COUPE, etc., dont le meilleur algorithme connu est en $O^*(2^n)$, la résolution en $O^*(2^{cn})$, $c < 1$, pour l'un d'entre eux implique-t-elle la résolution en $O^*(2^{c'n})$, $c' < 1$ pour un (ou plusieurs) autre(s) ?
- Peut on, en assumant l'insolvabilité des problèmes précédents à mieux que $O^*(2^n)$, produire des bornes inférieures de complexité pour d'autres problèmes ?
- Peut-on construire une hiérarchie de classes par rapport à la résolution (exacte) ? Avec quels ingrédients ? A quoi ressemblerait-t-elle ?

J'ai fini ! ! !