Approximation polynomiale : fondements et techniques

Bruno Escoffier, LIP6, Sorbonne Université

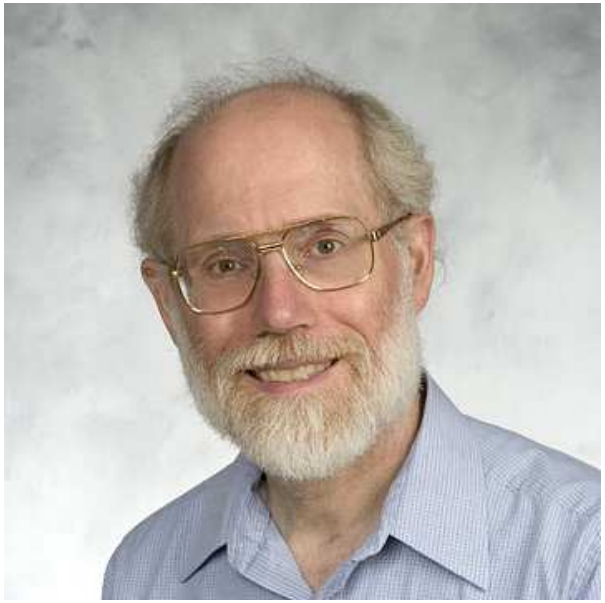
Journées Polyèdres et Optimisation Combinatoire

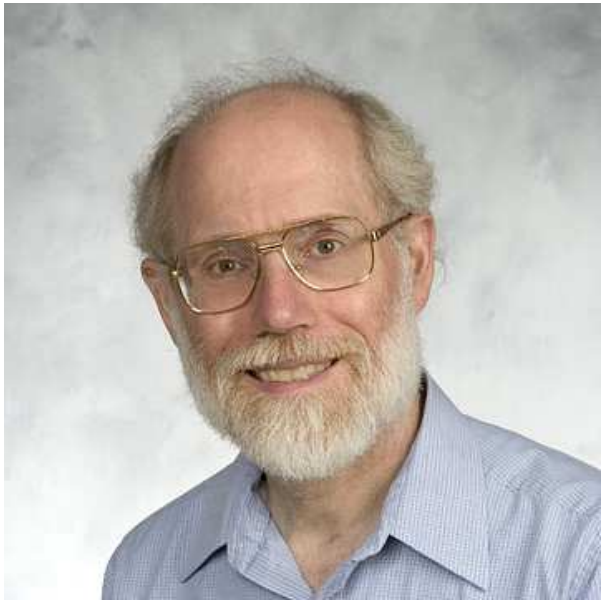
25 juin 2019

1. Année 0
2. Développements
3. Extensions du champ d'étude

Année 0

Année 0





David S. Johnson:

Approximation Algorithms for Combinatorial Problems. (STOC 1973, J. Comput. Syst. Sci. 1974)

Année -3: la *NP*-complétude

Année -3: la *NP*-complétude

Cook and Karp [NDLR and Levin] have shown the existence of a class of combinatorial problems, the "polynomial complete" problems, such that if any of these problems can be solved in polynomial time, then they all can.

D. Johnson, JCSS 74

Since no fast algorithms for finding an optimal solution could be found, simple heuristics algorithms that seemed likely to generate fairly good packings were studied.

D. Johnson, JCSS 74

Since no fast algorithms for finding an optimal solution could be found, simple heuristics algorithms that seemed likely to generate fairly good packings were studied.

1. Question existentielle: qu'est-ce qu'une solution "fairly good"?

D. Johnson, JCSS 74

Since no fast algorithms for finding an optimal solution could be found, simple heuristics algorithms that seemed likely to generate fairly good packings were studied.

1. Question existentielle: qu'est-ce qu'une solution "fairly good"? valeur "proche" de l'optimum?

D. Johnson, JCSS 74

Since no fast algorithms for finding an optimal solution could be found, simple heuristics algorithms that seemed likely to generate fairly good packings were studied.

1. Question existentielle: qu'est-ce qu'une solution "fairly good"? valeur "proche" de l'optimum?
2. Question pratique: comment prouver qu'on trouve une solution de valeur proche d'une valeur inconnue??

max independent set

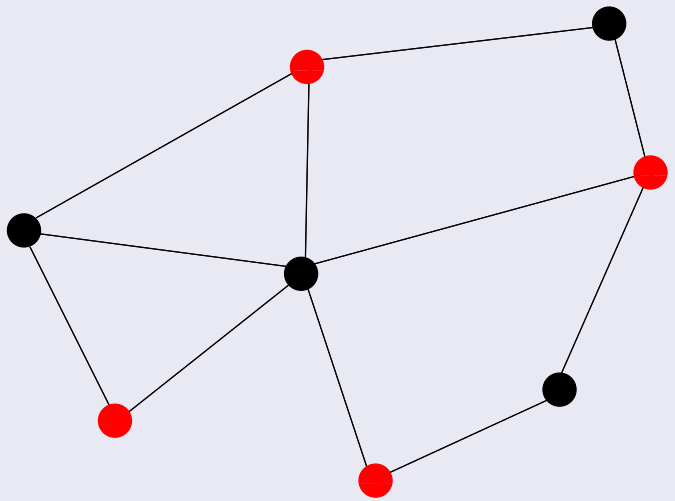


Figure: Indep. set: ensemble de sommets non adjacents

max independent set

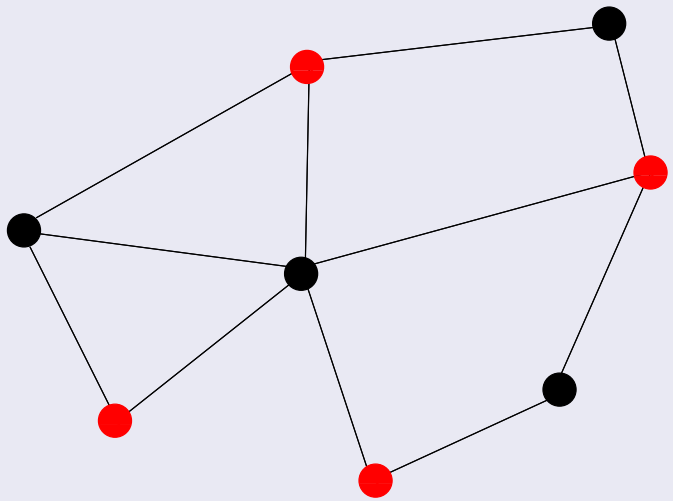


Figure: Indep. set: ensemble de sommets non adjacents

et sur son plus proche cousin: min vertex cover
Vertex Cover: ensemble de sommets 'touchant' toutes les arêtes.

max independent set

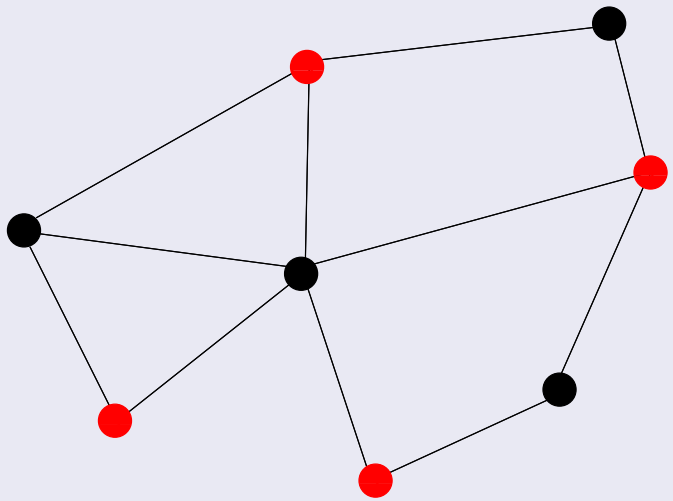


Figure: Indep. set: ensemble de sommets non adjacents

et sur son plus proche cousin: min vertex cover
Vertex Cover: ensemble de sommets 'touchant' toutes les arêtes.

S independent set $\Leftrightarrow$ $VC = V \setminus S$ vertex cover

max independent set

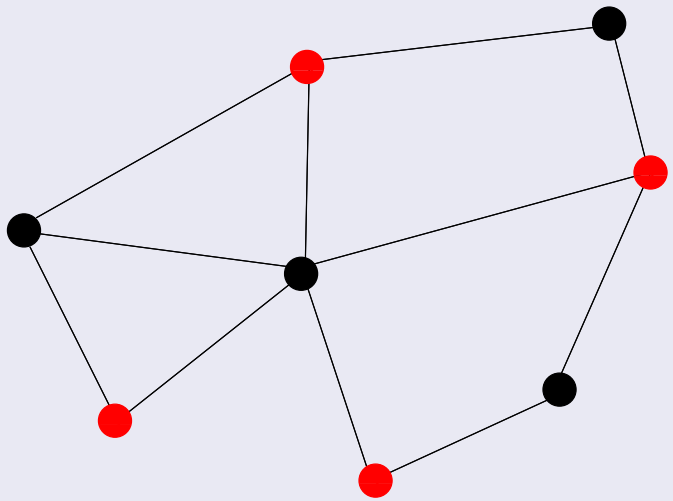


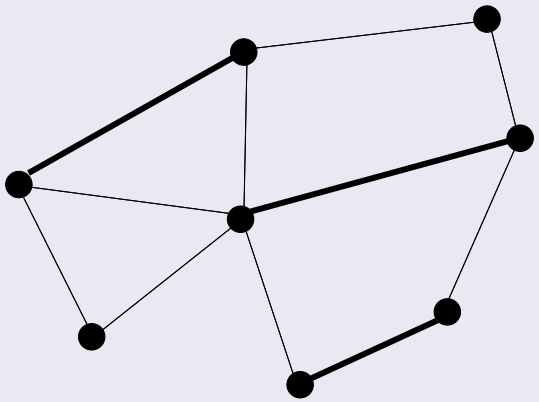
Figure: Indep. set: ensemble de sommets non adjacents

et sur son plus proche cousin: min vertex cover
Vertex Cover: ensemble de sommets 'touchant' toutes les arêtes.

S independent set $\Leftrightarrow$ $VC = V \setminus S$ vertex cover
Problème combinatoire: 2^n sous-ensembles de sommets.

Vertex Cover: un premier algorithme approché

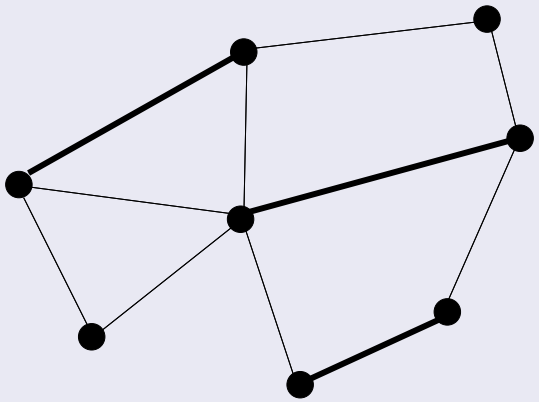
Prendre un ensemble maximal d'arêtes non adjacentes



Solution: VC =extrémités de ces arêtes

Vertex Cover: un premier algorithme approché

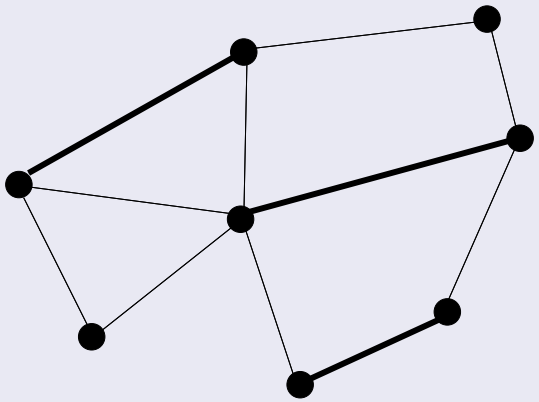
Prendre un ensemble maximal d'arêtes non adjacentes



Solution: VC =extrémités de ces arêtes
$\rightarrow$ Solution réalisable

Vertex Cover: un premier algorithme approché

Prendre un ensemble maximal d'arêtes non adjacentes



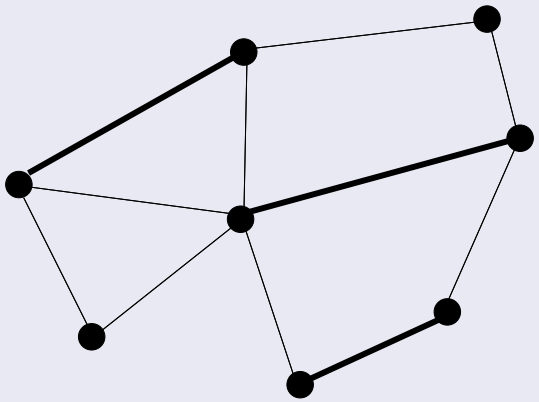
Solution: VC =extrémités de ces arêtes

$\rightarrow$ Solution réalisable

$\rightarrow |VC| = 2|M|$

Vertex Cover: un premier algorithme approché

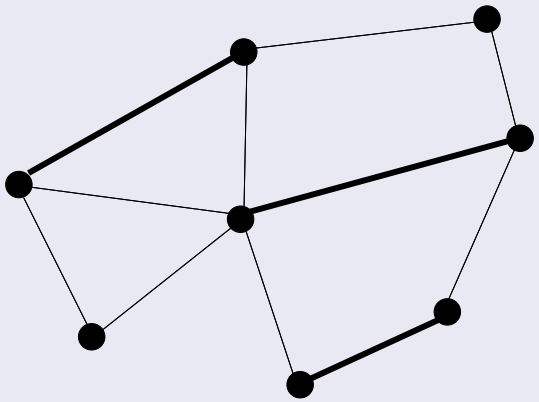
Prendre un ensemble maximal d'arêtes non adjacentes



Solution: VC =extrémités de ces arêtes
$\rightarrow$ Solution réalisable
$\rightarrow |VC| = 2|M|$
$|M| \leq \texttt{opt}(G)$

Vertex Cover: un premier algorithme approché

Prendre un ensemble maximal d'arêtes non adjacentes



Solution: VC =extrémités de ces arêtes
$\rightarrow$ Solution réalisable
$\rightarrow |VC| = 2|M|$
$|M| \leq \texttt{opt}(G)$
$\rightarrow |VC| \leq 2\texttt{opt}(G)$.

Algorithmes approchés

A est r-approché si pour toute instance I:

$$val(\mathtt{A}(I)) \leq r \times \mathrm{opt}(I)$$

$r \geq 1$ ($r = 1$: algorithme exact).

Algorithmes approchés

A est r-approché si pour toute instance I:

$$val(\mathtt{A}(I)) \leq r \times \mathrm{opt}(I)$$

$r \geq 1$ ($r = 1$: algorithme exact).

Définition analogue pour un problème de maximisation:

$$val(\mathtt{A}(I)) \geq r \times \mathrm{opt}(I)$$

$r \leq 1$ ($r = 1$: algorithme exact).

D. Johnson, JCSS 74

We discover that a wide variety of worst case behaviors are possible.

D. Johnson, JCSS 74

We discover that a wide variety of worst case behaviors are possible.

- Subset Sum
- Max Sat
- Set Cover
- Coloring
- Independent Set

Max Sat

Définition

- Un ensemble de n variables boolénnes, un ensemble de m clauses $((x_1 \vee \overline{x_3} \vee x_7))$.
- But: trouver une valeur de vérité qui satisfait un nombre maximal de clauses.

Max Sat

Définition

- Un ensemble de n variables boolénnes, un ensemble de m clauses $((x_1 \vee \overline{x_3} \vee x_7))$.
- But: trouver une valeur de vérité qui satisfait un nombre maximal de clauses.

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.

Max Sat

Définition

- Un ensemble de n variables boolénnes, un ensemble de m clauses $((x_1 \vee \overline{x_3} \vee x_7))$.
- But: trouver une valeur de vérité qui satisfait un nombre maximal de clauses.

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Toute clause est satisfaite par (au moins) une des deux solutions.

Max Sat

Définition

- Un ensemble de n variables boolénnes, un ensemble de m clauses $((x_1 \vee \overline{x_3} \vee x_7))$.
- But: trouver une valeur de vérité qui satisfait un nombre maximal de clauses.

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Toute clause est satisfaite par (au moins) une des deux solutions.
- $val(A(I)) \geq m/2 \geq opt(I)/2$

Algorithme 1/2-approché.

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Algorithme 1/2-approché.

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Algorithme 1/2-approché.

$\rightarrow$ Max Sat approximable à rapport constant: APX
$\rightarrow$ Peut-on améliorer ce rapport 1/2? 3/4? 0.99?

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Algorithme 1/2-approché.

$\rightarrow$ Max Sat approximable à rapport constant: APX
$\rightarrow$ Peut-on améliorer ce rapport 1/2? 3/4? 0.99?

- Avec notre algorithme?

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Algorithme 1/2-approché.

$\rightarrow$ Max Sat approximable à rapport constant: APX
$\rightarrow$ Peut-on améliorer ce rapport 1/2? 3/4? 0.99?

- Avec notre algorithme? Non: $(x_1 \vee x_2), (\overline{x_1} \vee \overline{x_2}) \rightarrow$ Notre analyse est optimale

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Algorithme 1/2-approché.

$\rightarrow$ Max Sat approximable à rapport constant: APX
$\rightarrow$ Peut-on améliorer ce rapport 1/2? 3/4? 0.99?

- Avec notre algorithme? Non: $(x_1 \vee x_2), (\overline{x_1} \vee \overline{x_2}) \rightarrow$ Notre analyse est optimale
- Avec un autre algorithme?

True/False

- Mettre tout à True, tout à False, et prendre la meilleure des deux solutions.
- Algorithme 1/2-approché.

$\rightarrow$ Max Sat approximable à rapport constant: APX
$\rightarrow$ Peut-on améliorer ce rapport 1/2? 3/4? 0.99?

- Avec notre algorithme? Non: $(x_1 \vee x_2), (\overline{x_1} \vee \overline{x_2}) \rightarrow$ Notre analyse est optimale
- Avec un autre algorithme?

$\rightarrow 1 - \epsilon, \forall \epsilon > 0$???

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1 - \epsilon)$-approché.

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1 - \epsilon)$-approché.

$\rightarrow$ Subset Sum polynomial?

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1-\epsilon)$-approché.

$\rightarrow$ Subset Sum polynomial? A_ϵ en $n^{1/\epsilon}$ par exemple.

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1-\epsilon)$-approché.

$\rightarrow$ Subset Sum polynomial? A_ϵ en $n^{1/\epsilon}$ par exemple.

$\rightarrow$ Classe *PTAS*.

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1-\epsilon)$-approché.

$\rightarrow$ Subset Sum polynomial? A_ϵ en $n^{1/\epsilon}$ par exemple.

$\rightarrow$ Classe *PTAS*.

$\rightarrow$ Mieux?

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1-\epsilon)$-approché.

$\rightarrow$ Subset Sum polynomial? A_ϵ en $n^{1/\epsilon}$ par exemple.

$\rightarrow$ Classe *PTAS*.

$\rightarrow$ Mieux?

$2^{1/\epsilon} n^3$ (Efficient PTAS)

Subset Sum

Schéma d'approximation

$\forall \epsilon > 0$, Subset Sum admet un algorithme $(1-\epsilon)$-approché.

$\rightarrow$ Subset Sum polynomial? A_ϵ en $n^{1/\epsilon}$ par exemple.

$\rightarrow$ Classe *PTAS*.

$\rightarrow$ Mieux?

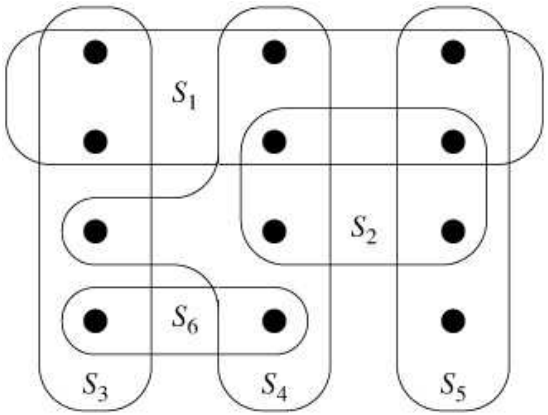
$2^{1/\epsilon} n^3$ (Efficient PTAS)

$1/\epsilon^3 n^2$ (Fully PTAS)

Set Cover

Définition

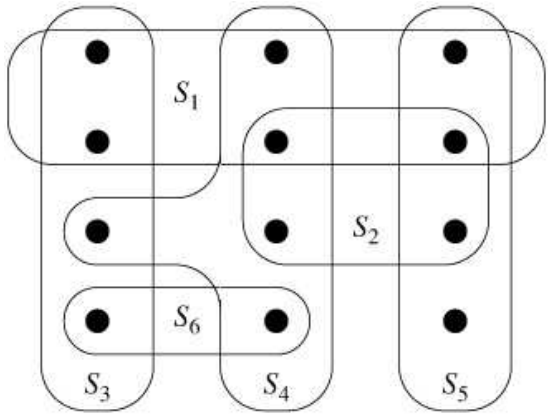
- Un ensemble $\mathcal{C}$, un ensemble de sous-ensembles $\mathcal{S} = (S_1, \ldots, S_m)$ de $\mathcal{C}$.
- But: couvrir $\mathcal{C}$ en utilisant un nombre minimum de sous-ensembles de $\mathcal{S}$.



Set Cover

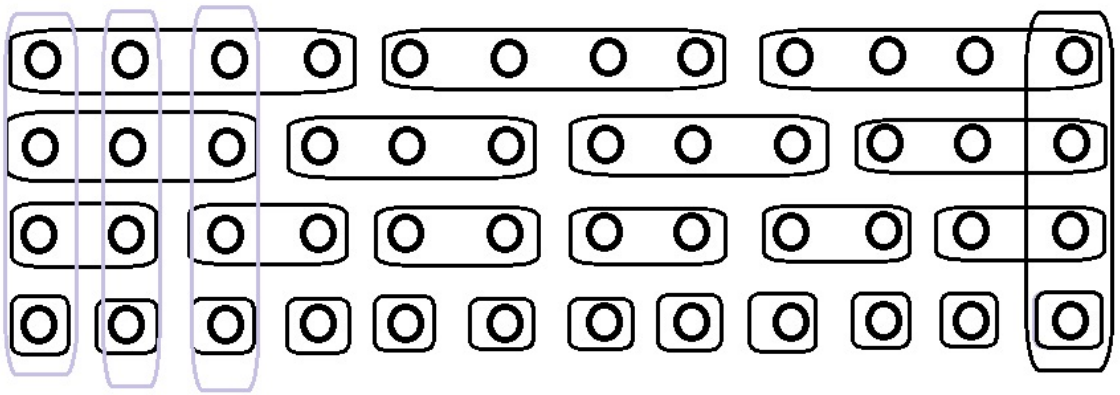
Définition

- Un ensemble $\mathcal{C}$, un ensemble de sous-ensembles $\mathcal{S} = (S_1, \ldots, S_m)$ de $\mathcal{C}$.
- But: couvrir $\mathcal{C}$ en utilisant un nombre minimum de sous-ensembles de $\mathcal{S}$.



- Algorithme glouton: prendre l'ensemble qui couvre le plus d'éléments.
- Rapport d'approximation?

Set Cover



Set Cover

Sur cette instance:

- $opt =$

Set Cover

Sur cette instance:

- $opt = t = 12$ (ensembles verticaux)
- Solution goutonne (si pas de chance)?

Set Cover

Sur cette instance:

- $opt = t = 12$ (ensembles verticaux)
- Solution goutonne (si pas de chance)?
  $t/4 + t/3 + t/2 + t = t(1 + 1/2 + 1/3 + 1/4)$

Set Cover

Sur cette instance:

- $opt = t = 12$ (ensembles verticaux)
- Solution goutonne (si pas de chance)?
  $t/4 + t/3 + t/2 + t = t(1 + 1/2 + 1/3 + 1/4)$
- Généralisation avec des ensembles de taille jusqu'a s: rapport
  $(1 + 1/2 + \cdots + 1/s) \rightarrow$ diverge!

Set Cover

Sur cette instance:

- $opt = t = 12$ (ensembles verticaux)
- Solution goutonne (si pas de chance)?
  $t/4 + t/3 + t/2 + t = t(1 + 1/2 + 1/3 + 1/4)$
- Généralisation avec des ensembles de taille jusqu'a s: rapport $(1 + 1/2 + \cdots + 1/s) \rightarrow$ diverge!
- Rapport non constant. Borne en fonction de paramètres de l'intense?
- Rapport $O(\log(|C|))$.

Set Cover

Sur cette instance:

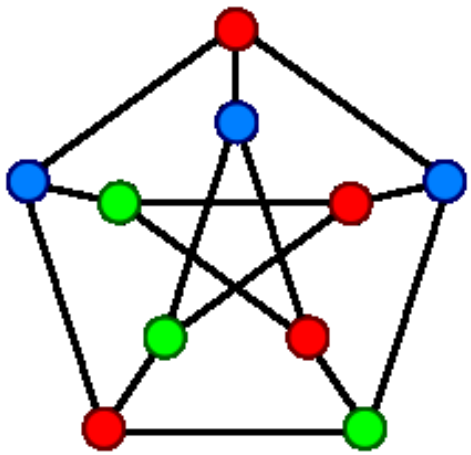
- $opt = t = 12$ (ensembles verticaux)
- Solution goutonne (si pas de chance)?
  $t/4 + t/3 + t/2 + t = t(1 + 1/2 + 1/3 + 1/4)$
- Généralisation avec des ensembles de taille jusqu'a s: rapport $(1 + 1/2 + \cdots + 1/s) \rightarrow$ diverge!
- Rapport non constant. Borne en fonction de paramètres de l'intense?
- Rapport $O(\log(|C|))$.
- Rapport constant? Schéma d'approximation?

Coloring et independent set

Coloring

$\rightarrow$ Attribuer une couleur à chaque sommet de manière à ce que 2 sommets adjacents n'aient jamais la même couleur.

$\rightarrow$ But: minimiser le nombre de couleurs.

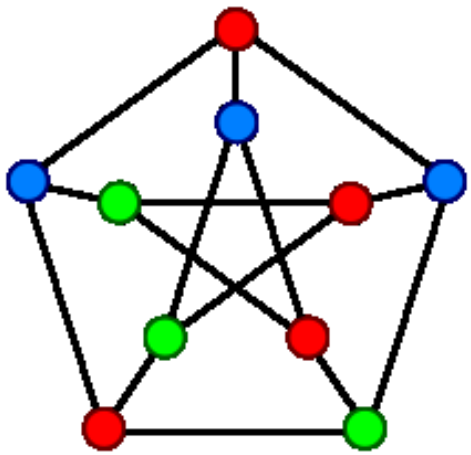


Coloring et independent set

Coloring

$\rightarrow$ Attribuer une couleur à chaque sommet de manière à ce que 2 sommets adjacents n'aient jamais la même couleur.

$\rightarrow$ But: minimiser le nombre de couleurs.



Coloring et max independent set

$\rightarrow$ Schéma?

$\rightarrow$ Approximable à rapport constant?

$\rightarrow$ Approximable à rapport $O(\log(n))$? $O(n^{\epsilon})$?

D. Johnson, JCSS 74

We discover that a wide variety of worst case behaviors are possible.

D. Johnson, JCSS 74

We discover that a wide variety of worst case behaviors are possible.

- Subset Sum: PTAS, $1 - \epsilon \ \forall \epsilon > 0$
- Max Sat: APX, 1/2-approché
- Set Cover: pas de rapport constant? rapport $\log(|C|)$
- Coloring: pas de rapport constant? rapport n
- Independent Set: pas de rapport constant? rapport $1/n$

Année 0: un programme de recherche

The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. [...] Many questions can be asked:

Année 0: un programme de recherche

The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. [...] Many questions can be asked:

- Are there indeed $O(\log n)$ coloring algorithms? Are there any clique finding algorithms better than $O(n^{\epsilon})$ for all ϵ?

Année 0: un programme de recherche

The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. [...] Many questions can be asked:

- Are there indeed $O(\log n)$ coloring algorithms? Are there any clique finding algorithms better than $O(n^{\epsilon})$ for all ϵ?
- Where do other optimization problems fit into the scheme of things?

Année 0: un programme de recherche

The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. [...] Many questions can be asked:

- Are there indeed $O(\log n)$ coloring algorithms? Are there any clique finding algorithms better than $O(n^{\epsilon})$ for all ϵ?
- Where do other optimization problems fit into the scheme of things?
- What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them?

Année 0: un programme de recherche

The results described in this paper indicate a possible classification of optimization problems as to the behavior of their approximation algorithms. [...] Many questions can be asked:

- Are there indeed $O(\log n)$ coloring algorithms? Are there any clique finding algorithms better than $O(n^{\epsilon})$ for all ϵ?
- Where do other optimization problems fit into the scheme of things?
- What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them?
- And what other types of behavior and ways of analyzing and measuring it are possible?

1. Année 0
2. Développements
   - Nombreuses techniques, innombrables résultats
   - Résultats négatifs
3. Quelques extensions du champ d'étude

Développements

- Nombreuses techniques, innombrables résultats: concevoir et analyser des algorithmes garantissant de bons rapports
- Résultats négatifs: jusqu'où peut-on aller??

Nombreuses techniques, innombrables résultats

- Algorithmes gloutons (Set Cover)
- Programmation dynamique (+ rounding, Knapsack)
- Recherche locale
- Utilisation de la programmation mathématique
- ...
- Algorithme spécifique

Définition de Max Cut

- Un graphe $G = (V, E)$
- Partitionner V en (V', V'') de manière à **maximiser** le nombre d'arêtes liant V' à V''.

Recherche locale

Définition de Max Cut

- Un graphe $G = (V, E)$
- Partitionner V en (V', V'') de manière à **maximiser** le nombre d'arêtes liant V' à V''.

Algorithme

- Partir d'une solution quelconque;
- Tant qu'on peut améliorer la solution en déplaçant un sommet: le faire;
- Renvoyer la solution obtenue.

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$
- $|E(V_1, V_2)| \geq 2|E(V_1)|$

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$
- $|E(V_1, V_2)| \geq 2|E(V_1)|$
- $|E(V_1, V_2)| \geq 2|E(V_2)|$

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$
- $|E(V_1, V_2)| \geq 2|E(V_1)|$
- $|E(V_1, V_2)| \geq 2|E(V_2)|$
- $|E(V_1, V_2)| \geq |E(V_1)| + |E(V_2)|$

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$
- $|E(V_1, V_2)| \geq 2|E(V_1)|$
- $|E(V_1, V_2)| \geq 2|E(V_2)|$
- $|E(V_1, V_2)| \geq |E(V_1)| + |E(V_2)|$
- $|E(V_1, V_2)| \geq |E|/2 \geq \text{opt}(G)/2$

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$
- $|E(V_1, V_2)| \geq 2|E(V_1)|$
- $|E(V_1, V_2)| \geq 2|E(V_2)|$
- $|E(V_1, V_2)| \geq |E(V_1)| + |E(V_2)|$
- $|E(V_1, V_2)| \geq |E|/2 \geq \text{opt}(G)/2$
- Algorithme polynomial?

Analyse de l'algorithme

Soit (V_1, V_2) la solution renvoyée.
Notation: $d_{V_1}(v) =$ nombres de voisins de v dans V_1.

- $v \in V_1$: $d_{V_2}(v) \geq d_{V_1}(v)$
- $\sum_{v \in V_1} d_{V_2}(v) \geq \sum_{v \in V_1} d_{V_1}(v)$
- $|E(V_1, V_2)| \geq 2|E(V_1)|$
- $|E(V_1, V_2)| \geq 2|E(V_2)|$
- $|E(V_1, V_2)| \geq |E(V_1)| + |E(V_2)|$
- $|E(V_1, V_2)| \geq |E|/2 \geq \text{opt}(G)/2$
- Algorithme polynomial?

L'analyse est optimale

Prendre un cycle sur 4 sommets, et une solution avec 2 sommets consécutifs.

Nombreuses techniques, innombrables résultats

- Algorithmes gloutons (Set Cover)
- Programmation dynamique (+ rounding, schéma pour Knapsack)
- Recherche locale
- Utilisation de la programmation mathématique
- ...
- Algorithme spécifique

PL et vertex cover

Vertex cover sous forme de PLNE

$$
\begin{aligned}
& Min \sum_{i=1}^{n} x_i \\
s.c. \quad & x_i + x_j \geq 1, \forall (i,j) \in E \\
& x_i \in \{0,1\}, i = 1, \ldots, n
\end{aligned}
$$

Relaxation

$$
\begin{aligned}
& Min \sum_{i=1}^{n} x_i \\
s.c. \quad & x_i + x_j \geq 1, \forall (i,j) \in E \\
& x_i \in [0,1], i = 1, \ldots, n
\end{aligned}
$$

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

PL et vertex cover

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

PL et vertex cover

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

Algorithme

- Résoudre le PL, trouver x^*
- Renvoyer la solution $\hat{x}$ avec $\hat{x}_i = \lceil x_i^* \rceil$

PL et vertex cover

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

Algorithme

- Résoudre le PL, trouver x^*
- Renvoyer la solution $\hat{x}$ avec $\hat{x}_i = \lceil x_i^* \rceil$

Analyse

- $\hat{x}_i \leq 2x_i^*$

PL et vertex cover

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

Algorithme

- Résoudre le PL, trouver x^*
- Renvoyer la solution $\hat{x}$ avec $\hat{x}_i = \lceil x_i^* \rceil$

Analyse

- $\hat{x}_i \leq 2x_i^*$ donc $val(\hat{x}) \leq 2val(x^*) \leq 2\text{opt}(G)$

PL et vertex cover

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

Algorithme

- Résoudre le PL, trouver x^*
- Renvoyer la solution $\hat{x}$ avec $\hat{x}_i = \lceil x_i^* \rceil$

Analyse

- $\hat{x}_i \leq 2x_i^*$ donc $val(\hat{x}) \leq 2val(x^*) \leq 2\text{opt}(G)$
- $\hat{x}$ est réalisable.
  Soit (i, j) une arête. $x_i^* + x_j^* \geq 1$, donc $\hat{x}_i = 1$ ou $\hat{x}_j = 1$.

PL et vertex cover

Théorème

Pour tout point extrême x^* du polyèdre défini par les contraintes ci-dessus, $x_i^* \in \{0, 1/2, 1\}$ pour tout i.

Algorithme

- Résoudre le PL, trouver x^*
- Renvoyer la solution $\hat{x}$ avec $\hat{x}_i = \lceil x_i^* \rceil$

Analyse

- $\hat{x}_i \leq 2x_i^*$ donc $val(\hat{x}) \leq 2val(x^*) \leq 2\text{opt}(G)$
- $\hat{x}$ est réalisable.
  Soit (i, j) une arête. $x_i^* + x_j^* \geq 1$, donc $\hat{x}_i = 1$ ou $\hat{x}_j = 1$.

Approximation et programmation linéaire

Par Nguyen Kim Thang.

Nombreuses techniques, innombrables résultats

- Algorithmes gloutons (Set Cover)
- Programmation dynamique (+ rounding, schéma pour Knapsack)
- Recherche locale
- Utilisation de la programmation mathématique
- ...
- Algorithme spécifique

Definition

- Un graphe complet dont les arêtes sont valuées.
- But: trouver un cycle passant une et une seule fois par chaque sommet de longueur totale minimale.
- Hypothèse: inégalité triangulaire: $d(x,z) \leq d(x,y) + d(y,z)$.

Definition

- Un graphe complet dont les arêtes sont valuées.
- But: trouver un cycle passant une et une seule fois par chaque sommet de longueur totale minimale.
- Hypothèse: inégalité triangulaire: $d(x,z) \leq d(x,y) + d(y,z)$.

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T sur le graphe.
- On faisant le tour de T, on a un cycle C passant par chaque sommet.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet.

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'.

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.
- $d(T) \leq opt(G)$.

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.
- $d(T) \leq opt(G)$.

- On peut faire mieux... (1.5)

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.
- $d(T) \leq opt(G)$.

- On peut faire mieux... (1.5)
- Dans le cas Euclidien: schema d'approximation, Arora et Mitchell (Prix Gödel 36)

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.
- $d(T) \leq opt(G)$.

- On peut faire mieux... (1.5)
- Dans le cas Euclidien: schema d'approximation, Arora et Mitchell (Prix Gödel 36)
- PM: Max Cut 0.878-approche avec de la programmation semi-definie, Goemans et Williamson (Prix Fulkerson 26)

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.
- $d(T) \leq opt(G)$.

- On peut faire mieux... (1.5)
- Dans le cas Euclidien: schema d'approximation, Arora et Mitchell (Prix Gödel 36)
- PM: Max Cut 0.878-approche avec de la programmation semi-definie, Goemans et Williamson (Prix Fulkerson 26)
- Vertex Cover: mieux que 2?

Un algorithme 2-approché

- On sait connecter les sommets a faible coût: MST T.
- On faisant le tour de T, on a un cycle C passant par chaque sommet. $d(C) = 2d(T)$.
- Si on passe par un sommet deja visité, on le saute (raccourcis) $\rightarrow$ solution C'. $d(C') \leq d(C) = 2d(T)$.
- $d(T) \leq opt(G)$.

- On peut faire mieux... (1.5)
- Dans le cas Euclidien: schema d'approximation, Arora et Mitchell (Prix Gödel 36)
- PM: Max Cut 0.878-approche avec de la programmation semi-definie, Goemans et Williamson (Prix Fulkerson 26)
- Vertex Cover: mieux que 2? Et les autres problèmes de l'article de Johnosn?

Résultats négatifs

Résultats négatifs

Questions de Johnson (rappel)

- Algorithme à rapport constant pour coloring? pour independent set? pour set cover?
- Schéma d'approximation pour ces problèmes? Pour Sat?

Résultats négatifs

Questions de Johnson (rappel)

- Algorithme à rapport constant pour coloring? pour independent set? pour set cover?
- Schéma d'approximation pour ces problèmes? Pour Sat?

$\rightarrow$ première approche, premiers résultats
$\rightarrow$ le théorème PCP

Approche par réductions

Année 0

Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results

Approche par réductions

Année 0

Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results

Premiers résultats négatifs

- 3-coloring est *NP*-complet.

Approche par réductions

Année 0

Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results

Premiers résultats négatifs

- 3-coloring est NP-complet.
- Pour tout $\epsilon > 0$, Coloring n'est pas approximable à rapport $4/3 - \epsilon$ si $P \neq NP$.

Approche par réductions

Année 0

Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results

Premiers résultats négatifs

- 3-coloring est NP-complet.
- Pour tout $\epsilon > 0$, Coloring n'est pas approximable à rapport $4/3 - \epsilon$ si $P \neq NP$.
  Si algorithme A $(4/3 - \epsilon)$-approché: soit G un graphe.
  - Si $A(G)$ utilise au plus 3 couleurs, G est 3-coloriable.
  - Si $A(G)$ utilise au moins 4 couleurs: $\texttt{opt(G)} \leq 3$?

Approche par réductions

Année 0

Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results

Premiers résultats négatifs

- 3-coloring est NP-complet.
- Pour tout $\epsilon > 0$, Coloring n'est pas approximable à rapport $4/3 - \epsilon$ si $P \neq NP$.
  Si algorithme A $(4/3 - \epsilon)$-approché: soit G un graphe.
  - Si $A(G)$ utilise au plus 3 couleurs, G est 3-coloriable.
  - Si $A(G)$ utilise au moins 4 couleurs: $\texttt{opt(G)} \leq 3$?
    $4 \leq A(G) \leq (4/3 - \epsilon)\texttt{opt(G)} \leq 4 - 3\epsilon < 4$.

Approche par réductions

Année 0

Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results

Premiers résultats négatifs

- 3-coloring est *NP*-complet.
- Pour tout $\epsilon > 0$, Coloring n'est pas approximable à rapport $4/3 - \epsilon$ si $P \neq NP$.
  Si algorithme A $(4/3 - \epsilon)$-approché: soit G un graphe.
  - Si $A(G)$ utilise au plus 3 couleurs, G est 3-coloriable.
  - Si $A(G)$ utilise au moins 4 couleurs: $\texttt{opt(G)} \leq 3$?
    $4 \leq A(G) \leq (4/3 - \epsilon)\texttt{opt(G)} \leq 4 - 3\epsilon < 4$.
- Plus généralement: partir d'un problème *NP*-complet et créer un saut dans le problème d'arrivée $\rightarrow$ inapproximabilité.

Approche par réductions

Réduction préservant l'approximation

$A \leq B$: si B est approximable à ratio ..., alors A est approximable à ratio ...

Approche par réductions

Réduction préservant l'approximation

$A \leq B$: si B est approximable à ratio ..., alors A est approximable à ratio ...

Résultats négatifs si $P \neq NP$

- Des résultats d'inapproximabilité pour certains problèmes
  $\rightarrow$ Coloring n'admet pas de schéma
- Quid de nos autres questions?

Approche par réductions

Réduction préservant l'approximation

$A \leq B$: si B est approximable à ratio ..., alors A est approximable à ratio ...

Résultats négatifs si $P \neq NP$

- Des résultats d'inapproximabilité pour certains problèmes
  $\rightarrow$ Coloring n'admet pas de schéma
- Quid de nos autres questions?

Une réponse partielle: des équivalences entre problèmes

Problèmes Max-SNP complets, Papadimitriou Yannakakis 14: Max-SAT-B admet un schéma ssi IS-B admet un schéma, ssi VC-B admet un schéma,...

Résultats négatifs

Le théorème PCP, année 18

Une nouvelle caractérisation de *NP*, et ses conséquences pour l'approximation polynomiale.

2 Prix Gödel:

- 27: Sanjeev Arora, Uriel Feige, Shafi Goldwasser, Carsten Lund, László Lovász, Rajeev Motwani, Shmuel Safra, Madhu Sudan et Mario Szegedy
- 37: Johan Hastad

Résultats négatifs

Conséquences des systèmes PCP

Si $P \neq NP$:

- max independent set inapproximable avec un rapport constant, et même avec un rapport $1/n^c$ pour tout $c < 1$
- min coloring avec un rapport constant, et même avec un rapport n^c pour tout $c < 1$
- min set cover avec un rapport constant, et même avec un rapport $o(\log |\mathcal{C}|)$
- max-3-SAT n'est pas approximable avec un rapport meilleur que $7/8$
- ...

Résultats négatifs

Conséquences des systèmes PCP

Si $P \neq NP$:

- max independent set inapproximable avec un rapport constant, et même avec un rapport $1/n^c$ pour tout $c < 1$
- min coloring avec un rapport constant, et même avec un rapport n^c pour tout $c < 1$
- min set cover avec un rapport constant, et même avec un rapport $o(\log |\mathcal{C}|)$
- max-3-SAT n'est pas approximable avec un rapport meilleur que $7/8$
- ...

Remarque

Ces résultats matchent exactement les résultats (positifs) de l'article de Johnson!!

- Une vue assez précise du paysage, des outils puissants
- Un champ toujours très actif:
  - Des sauts parfois importants: TSP, TSP asymétrique!
  - De nouveaux problèmes
  - De nouveaux challenges

On the Traveling Salesperson Problem with Neighborhoods

Antonios Antoniadis

Scheduling problems

Several talks coming!

1. Année 0
2. Développements
3. Extensions du champ d'étude

- Changement de paradigme:
  - Modulation autour du temps de calcul
  - Une mesure alternative: l'approximation différentielle
- Prise en compte du contexte
  - Problèmes en environnement évolutifs ou incertains
  - Problèmes multicritères, multi-décideurs, multiagents

Changement de paradigme

On cherche des algorithmes:

- de complexité polynomiale
- ayant un bon rapport d'approximation.

Changement de paradigme

Complexité polynomiale?

- Un algorithme en n^5 pour des très gros graphes?

Changement de paradigme

Complexité polynomiale?

- Un algorithme en n^5 pour des très gros graphes?
- Un rapport $n^{\epsilon-1}$ impossible à garantir?
  $\rightarrow$ Compromis temps/qualité, au-delà du temps polynomial

Changement de paradigme

Complexité polynomiale?

- Un algorithme en n^5 pour des très gros graphes?
- Un rapport $n^{\epsilon-1}$ impossible à garantir?
  $\rightarrow$ Compromis temps/qualité, au-delà du temps polynomial

Résolution exacte ou approchée en temps exponentiel avec application à l'ordonnancement

Par Vincent T'Kindt.

Changement de paradigme

Rapport d'approximation

- S independent set ssi $V \setminus S$ est un vertex cover

Changement de paradigme

Rapport d'approximation

- S independent set ssi $V \setminus S$ est un vertex cover
- un algorithme A pour independent set n'a-t-il pas la même qualité qu'un algorithme renvoyant $V \setminus A(G)$ pour vertex cover???

Changement de paradigme

Rapport d'approximation

- S independent set ssi $V \setminus S$ est un vertex cover
- un algorithme A pour independent set n'a-t-il pas la même qualité qu'un algorithme renvoyant $V \setminus A(G)$ pour vertex cover???
  $\rightarrow$ Biais du rapport d'approximation "standard"

Changement de paradigme

Rapport d'approximation

- S independent set ssi $V \setminus S$ est un vertex cover
- un algorithme A pour independent set n'a-t-il pas la même qualité qu'un algorithme renvoyant $V \setminus A(G)$ pour vertex cover???
  $\rightarrow$ Biais du rapport d'approximation "standard"
  $\rightarrow$ Une autre mesure de la qualité d'une solution: le rapport différentiel

- Changement de paradigme:
  - Modulation autour du temps de calcul
  - Une mesure alternative: l'approximation différentielle
- Prise en compte du contexte
  - Problèmes en environnement évolutifs ou incertains
  - Problèmes multicritères, multi-décideurs, multiagents

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données
- Une fonction objectif claire

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données?

Données imprécises et/ou partiellement connues et/ou susceptibles d'évoluer

Prise en compte du contexte

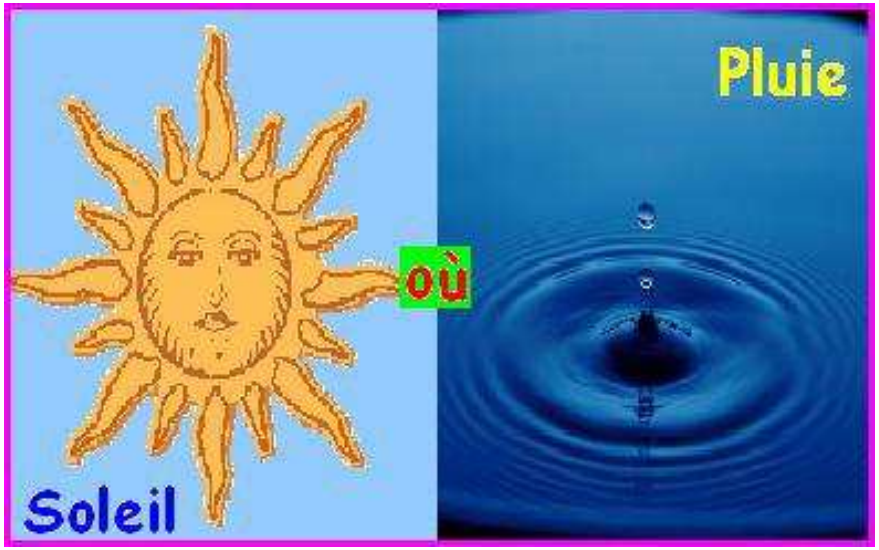
Cadre d'étude

- Une connaissance totale et parfaite des données?

Données imprécises et/ou partiellement connues et/ou susceptibles d'évoluer

$\rightarrow$ Optimisation robuste

Optimisation robuste



- Données imprécises: un ensemble de scenarios possibles / des intervalles de valeurs;
- But: trouver une solution *robuste*: bonne quel que soit le scenario qui se réalisera

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données?

Données imprécises et/ou partiellement connues et/ou susceptibles d'évoluer

→ Optimisation robuste

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données?

Données imprécises et/ou partiellement connues et/ou susceptibles d'évoluer

→ Optimisation robuste
→ Optimisation stochastique, optimisation combinatoire probabiliste
→ Optimisation on-line

Optimisation on-line

- Données révélées au fur et à mesure;
- Principe: construire une solution au fur et à mesure

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données?

Données imprécises et/ou partiellement connues et/ou susceptibles d'évoluer

→ Optimisation robuste
→ Optimisation stochastique, optimisation combinatoire probabiliste
→ Optimisation on-line
→ Réoptimisation

Réoptimisation

- Une instance déjà résolue légèrement perturbée;
- But: trouver une solution après perturbation.

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données?

Données imprécises et/ou partiellement connues et/ou susceptibles d'évoluer

→ Optimisation robuste
→ Optimisation stochastique, optimisation combinatoire probabiliste
→ Optimisation on-line
→ Réoptimisation
→ optimisation temporelle/multistage

Over-time optimization: positive and negative results

Evripidis Bampis

Prise en compte du contexte

Cadre d'étude

- Une connaissance totale et parfaite des données
- Une fonction objectif claire

Prise en compte du contexte

Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

Prise en compte du contexte

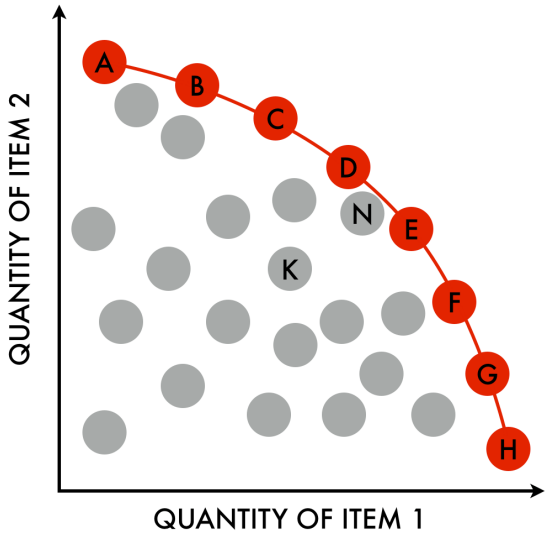
Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

$\rightarrow$ Optimisation multi-critère

Optimisation multi-critère



Prise en compte du contexte

Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

Prise en compte du contexte

Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

→ Optimisation multi-critère
→ Choix social combinatoire

Choix social combinatoire

- Trouver la meilleure alternative, aggréger des préférences individuelles

Prise en compte du contexte

Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

Prise en compte du contexte

Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

→ Optimisation multi-critère
→ Choix social combinatoire

Prise en compte du contexte

Cadre d'étude

- Une fonction objectif claire?

Problème multi-critère, multi-décideur, centralisé ou décentralisé

→ Optimisation multi-critère
→ Choix social combinatoire
→ Théorie des jeux algorithmiques

- Prix de l'anarchie
- Mécanismes de coordination

Bonne fin de journée!

Set Cover

Définition

- Un ensemble $\mathcal{C}$, un ensemble de sous-ensembles $\mathcal{S} = (S_1, \ldots, S_m)$ de $\mathcal{C}$.
- But: couvrir $\mathcal{C}$ en utilisant un nombre minimum de sous-ensembles de $\mathcal{S}$.

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, *opt* la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, opt la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).
- A l'itération i: m_{i-1} éléments à couvrir

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, opt la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).
- A l'itération i: m_{i-1} éléments à couvrir $\rightarrow$ un ensemble couvrant au moins $\frac{m_{i-1}}{opt}$ éléments

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, opt la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).
- A l'itération i: m_{i-1} éléments à couvrir $\rightarrow$ un ensemble couvrant au moins $\frac{m_{i-1}}{opt}$ éléments

  $\rightarrow m_i \leq m_{i-1} - \frac{m_{i-1}}{opt} = m_{i-1}\left(1 - \frac{1}{opt}\right)$

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, opt la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).
- A l'itération i: m_{i-1} éléments à couvrir $\rightarrow$ un ensemble couvrant au moins $\frac{m_{i-1}}{opt}$ éléments

  $\rightarrow m_i \leq m_{i-1} - \frac{m_{i-1}}{opt} = m_{i-1}\left(1 - \frac{1}{opt}\right)$

  $\rightarrow m_i \leq m_0\left(1 - \frac{1}{opt}\right)^i$

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, opt la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).
- A l'itération i: m_{i-1} éléments à couvrir $\rightarrow$ un ensemble couvrant au moins $\frac{m_{i-1}}{opt}$ éléments

  $\rightarrow m_i \leq m_{i-1} - \frac{m_{i-1}}{opt} = m_{i-1}\left(1 - \frac{1}{opt}\right)$

  $\rightarrow m_i \leq m_0\left(1 - \frac{1}{opt}\right)^i$
- $m_{g-1} \geq 1$ (sinon on se serait arrêté), donc

  $1 \leq m\left(1 - \frac{1}{opt}\right)^{g-1}$

Algorithme glouton

$\rightarrow$ Prendre l'ensemble qui a le plus d'éléments (parmi ceux non couverts).

- g la taille de la solution gloutonne, opt la taille d'une couverture optimale.
- m_i : nombre d'éléments restant à couvrir après i itérations ($m_0 = m = |\mathcal{C}|$).
- A l'itération i: m_{i-1} éléments à couvrir $\rightarrow$ un ensemble couvrant au moins $\frac{m_{i-1}}{opt}$ éléments

  $\rightarrow m_i \leq m_{i-1} - \frac{m_{i-1}}{opt} = m_{i-1}\left(1 - \frac{1}{opt}\right)$

  $\rightarrow m_i \leq m_0\left(1 - \frac{1}{opt}\right)^i$
- $m_{g-1} \geq 1$ (sinon on se serait arrêté), donc

  $1 \leq m\left(1 - \frac{1}{opt}\right)^{g-1}$

  $\rightarrow g - 1 \leq opt \ln(m)$ (en utilisant $(1 - 1/a)^a \leq 1/e$).

Définition

- Un ensemble $\{1, \ldots, n\}$ de n objets: l'objet i a une valeur $v_i \in \mathbb{N}$ et un poids $p_i \in \mathbb{N}$.
- Un poids maximum $B \in \mathbb{N}$.
- But: choisir un sous-ensemble d'objets S tq $\sum_{i \in S} p_i \leq B$ en maximisant $\sum_{i \in S} v_i$

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.
- Complexité: $V' \leq V/K = n^2/\epsilon$ donc $O(n^3/\epsilon)$.

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.
- Complexité: $V' \leq V/K = n^2/\epsilon$ donc $O(n^3/\epsilon)$.
- Rapport? $\frac{v_i}{K} - 1 \leq v'_i \leq \frac{v_i}{K}$

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.
- Complexité: $V' \leq V/K = n^2/\epsilon$ donc $O(n^3/\epsilon)$.
- Rapport? $\frac{v_i}{K} - 1 \leq v'_i \leq \frac{v_i}{K}$
  $\rightarrow \frac{v(S)}{K} - n \leq v'(S) \leq \frac{v(S)}{K}$.

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.
- Complexité: $V' \leq V/K = n^2/\epsilon$ donc $O(n^3/\epsilon)$.
- Rapport? $\frac{v_i}{K} - 1 \leq v'_i \leq \frac{v_i}{K}$
  $\rightarrow \frac{v(S)}{K} - n \leq v'(S) \leq \frac{v(S)}{K}$.
  $\rightarrow v(\hat{S}) \geq Kv'(\hat{S}) \geq Kv'(S^*) \geq v(S^*) - Kn$

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.
- Complexité: $V' \leq V/K = n^2/\epsilon$ donc $O(n^3/\epsilon)$.
- Rapport? $\frac{v_i}{K} - 1 \leq v'_i \leq \frac{v_i}{K}$
  $\rightarrow \frac{v(S)}{K} - n \leq v'(S) \leq \frac{v(S)}{K}$.
  $\rightarrow v(\hat{S}) \geq Kv'(\hat{S}) \geq Kv'(S^*) \geq v(S^*) - Kn$
  $Kn = \frac{\epsilon V}{n} \leq \epsilon v_{max} \leq \epsilon v(S^*)$

Algorithme pseudo-polynomial

$\rightarrow$ de complexité $O(nV)$, où $V = \sum_{i=1}^{n} v_i$.

Algorithme approché: arrondi

- Résoudre le problème avec des valeurs $v'_i = v_i/K$, avec $K = \epsilon V/n$ (poids et B inchangé) $\rightarrow$ solution $\hat{S}$.
- Complexité: $V' \leq V/K = n^2/\epsilon$ donc $O(n^3/\epsilon)$.
- Rapport? $\frac{v_i}{K} - 1 \leq v'_i \leq \frac{v_i}{K}$
  $\rightarrow \frac{v(S)}{K} - n \leq v'(S) \leq \frac{v(S)}{K}$.
  $\rightarrow v(\hat{S}) \geq Kv'(\hat{S}) \geq Kv'(S^*) \geq v(S^*) - Kn$
  $Kn = \frac{\epsilon V}{n} \leq \epsilon v_{max} \leq \epsilon v(S^*)$
  $\rightarrow v(\hat{S}) \geq (1-\epsilon)v(S^*)$.