Les enchères et les mécanismes véraces

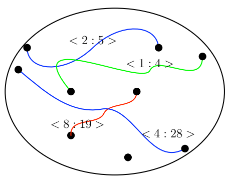
Johanne Cohen

LORIA/CNRS, Nancy, France.

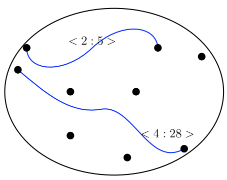
Réservation des ressources (path auctions)

- Un réseau : $G = (V, E)$ avec capacités $c : E \to \mathbb{R}$
- Un ensemble de requêtes $\mathcal{R}$ définit $\langle chemin, demande, budget \rangle$
- Un administrateur.

Objectif global : affecter le plus grand nombre de ressources en maximisant le bénéfice de l'administrateur (coût social).



réseau avec un ens. de requêtes.



réseau avec un ens. de requêtes sélectionnées.

Analogie avec les enchères combinatoires

Enchère combinatoire (Allocation des ressources)

Énoncé

- un ens. de m objets U (de m arêtes E)
- 1 ens. de n participants (requêtes) (N)
- Chaque participant (requête) i a
  - une fct de valuation (privée) t_i pour chaque sous-ensemble $S \subseteq U$ (1 chemin de capacité c_i)
  - une fct de déclaration d_i pour chaque $S \subseteq U$ (pour ce chemin)

Objectif

- allouer les objets de façon à maximiser le *coût social* ($w = \sum_j d_j(S_j)$ avec S_j le lot affecté à j)
- déterminer le paiement pour chaque participant.

Comment convaincre les participants de révéler leur réelle demande et leur réelle valuation ?

- Trouver des mécanismes qui permettent d'allouer les chemins au bon prix.
- Par exemple : les mécanismes de VCG.

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - `Alice` : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - `Bob` : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ?

  Et à quel prix ?

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ?

  Et à quel prix ?

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ?

  Et à quel prix ?

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ?

  Et à quel prix ?

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ? celui qui a fait la meilleur annonce

  Et à quel prix ? sa déclaration

Scénario : Alice remporte l'objet au prix de 25

- Gain d' Alice : $u_{\texttt{Alice}} = 40 - 25 = 15$
- Gain de Bob : $u_{\texttt{Bob}} = 0$

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ? celui qui a fait la meilleur annonce

  Et à quel prix ? sa déclaration

Scénario : Alice remporte l'objet au prix de 25

- Gain d' Alice : $u_{\texttt{Alice}} = 40 - 25 = 15$
- Gain de Bob : $u_{\texttt{Bob}} = 0$

Illustration : un objet (ou une arête de capacité 1)

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 19$,
- 2 questions :

  Qui remporte ? celui qui a fait la meilleur annonce

  Et à quel prix ? sa déclaration

Scénario : Alice remporte l'objet au prix de 25

- Gain d' Alice : $u_{\texttt{Alice}} = 40 - 25 = 15$
- Gain de Bob : $u_{\texttt{Bob}} = 0$

Définition : mécanisme

Données

- un ensemble de m objets (U)
- un ensemble de n participants (N) avec les vecteurs $\vec{t}$ et $\vec{d}$

Un mécanisme est caractérisé par

1. un algorithme d'allocation x : $x_i(\vec{d})$ le lot affecté à i
2. une fonction de paiement p : le participant i doit honorer $p_i(\vec{d})$ pour le lot $x_i(\vec{d})$

L'utilité pour i :$u_i = \begin{cases} valuation - paiement & si \quad x_i(\vec{d}) \neq \emptyset \\ 0 & sinon \end{cases}$

Définition : mécanisme

Données

- un ensemble de m objets (U)
- un ensemble de n participants (N) avec les vecteurs $\vec{t}$ et $\vec{d}$

Un mécanisme est caractérisé par

1. un algorithme d'allocation x : $x_i(\vec{d})$ le lot affecté à i
2. une fonction de paiement p : le participant i doit honorer $p_i(\vec{d})$ pour le lot $x_i(\vec{d})$

L'utilité pour i : $u_i = \begin{cases} t(x_i(\vec{d})) - p_i(\vec{d}) & si \quad x_i(\vec{d}) \neq \emptyset \\ 0 & sinon \end{cases}$

Mécanisme véräce

Un mécanisme (x, p) est vérace si quelque soit le participant i, la stratégie t_i de "dire la vérité" est une meilleure stratégie, i.e :

$$\forall \vec{d} \text{ vecteur de déclarations, } u_i(t_i, d_{-i}) \geq u_i(d_i, d_{-i})$$

avec d_{-i} vecteur de déclarations de tous les participants sauf celui de i.

Retour à l'exemple : Enchère vérace ?

- 1 objet (U)
- 2 participants ($N = \{\texttt{Alice}, \texttt{Bob}\}$)
  - Alice : $t_{Alice} = 40$, $d_{Alice} = 25$,
  - Bob : $t_{Bob} = 24$, $d_{Bob} = 20$,
- 2 questions :
  Qui remporte ? Et à quel prix ?

Scénario : Alice remporte l'objet au prix de 25

Gain d' Alice : $u_{\texttt{Alice}} = 40 - 25 = 15$

Le mécanisme $\begin{cases} allocation = premier\ prix \\ prix = declaration \end{cases}$ n'est pas vérace.

Enchères de Vickrey pour un object

Données :

- n joueurs voulant acquérir un objet $\mathcal{O}$
- vecteur de type $\vec{t}$ et de déclarations $\vec{d}$

Objectif global (coût social) : déterminer un prix le plus haut possible.

Objectif local pour i : $u_i = \begin{cases} \text{maximiser } t_i - prix & \text{si } i \text{ achète } \mathcal{O} \\ 0 & \text{sinon} \end{cases}$

Description du mécanisme :

1. Algorithme d'allocation : celui qui a donné la meilleure offre.
2. Algorithme de paiement : la seconde meilleure offre.

$$prix = \max\{d_j : j \neq \arg\max\{d_j\}\}$$

Illustration

	valeur qu'il est prêt à mettre t_i	valeur qu'il déclare d_i	p_i	utilité $u_i = t_i - p_i$
Stratégie "*dire la vérité*" pour Alice				
joueur Alice	40	40	19	21
joueur Bob	24	19	0	0
Stratégie "*Sous-estimer le prix*" pour Alice				
joueur Alice	40	25	19	21
joueur Bob	24	19	0	0
Stratégie "*Sous-estimer le prix*" pour Alice				
joueur Alice	40	14	0	0
joueur Bob	24	19	14	10

Illustration

	valeur qu'il est prêt à mettre t_i	valeur qu'il déclare d_i	p_i	utilité $u_i = t_i - p_i$
Stratégie "*dire la vérité*" pour `Alice`				
`joueur Alice`	40	40	19	21
joueur `Bob`	24	19	0	0
Stratégie "*Sous-estimer le prix*" pour `Alice`				
`joueur Alice`	40	25	19	21
joueur `Bob`	24	19	0	0
Stratégie "*Sous-estimer le prix*" pour `Alice`				
`joueur Alice`	40	14	0	0
joueur `Bob`	24	19	14	10

Illustration

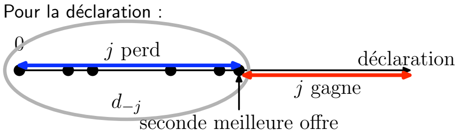
	valeur qu'il est prêt à mettre t_i	valeur qu'il déclare d_i	p_i	utilité $u_i = t_i - p_i$
Stratégie "*dire la vérité*" pour `Alice`				
`joueur Alice`	40	40	19	21
joueur `Bob`	24	19	0	0
Stratégie "*Sous-estimer le prix*" pour `Alice`				
`joueur Alice`	40	25	19	21
joueur `Bob`	24	19	0	0
Stratégie "*Sous-estimer le prix*" pour `Alice`				
`joueur Alice`	40	14	0	0
joueur `Bob`	24	19	14	10

Les enchères de Vickrey sont des mécanismes véraces

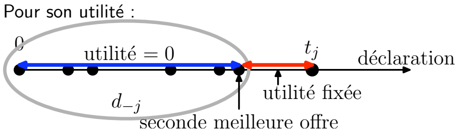
Theorem (Vickrey73)

Les mécanismes VCG sont des mécanismes véraces.

- Pour la déclaration :



- Pour son utilité :



Enchères de Vickrey généralisées (GVA).

Soit $\vec{d}$ le vecteur des déclarations.

- La fonction d'allocation est de maximiser la somme des valeurs déclarées pour les acheteurs.
- Chaque acheteur reçoit la prime correspondant à la somme correspondant au gain si il participe.

$prix_j(\vec{d})$ = valeur déclarée -(Valeur de l'ensemble avec LUI - Valeur de l'ensemble SANS LUI)

Theorem (Vickrey73)

Les mécanismes généralisées VCG sont des mécanismes véraces.

Résultat sur l'allocation de ressources

Enoncé

- Un réseau : $G = (V, E)$ avec capacités $c : E \rightarrow \mathbb{R}$
- Un ensemble de requêtes $\mathcal{R}$ définit $\langle$*chemin*, *demande*, *budget*$\rangle$

Objectif :

Trouver un mécanisme véarce $\langle$ algo d'allocation, fct de paiement $\rangle$

Quelque résultats sur l'algorithme d'allocation :

	1 arête	Chemin	Arbre	Graphe
requête unitaire	Poly (algo glouton)	Poly (algo dynamique)	NP-Complet 1/4-approx. [2003]	NP-Complet $O(1/\sqrt{n})$-approx [1999]
requête non-unitaire	NP-Complet 1/2-approx FPTAS	NP-Complet 1/3-approx. [2002]	NP-Complet 1/48-approx. [2003]	NP-Complet $O(1/\sqrt{n})$-approx [1999]

Mécanisme vérace : pour les cas polynomiaux

	1 arête	Chemin	Arbre	Graphe
requête unitaire	Poly mécanisme VCG	Poly mécanisme VCG		
requête non-unitaire				

Dans les problèmes où le problème d'affectation est polynomial, les enchères VCG généralisées peuvent s'appliquer.
Que se passe-t-il pour les cas non-polynomiaux ?

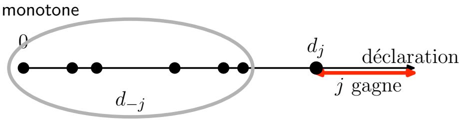
Enchère de Vickrey : propriété

Définition

La fonction d'allocation est monotone si pour chaque le participant j et pour $\forall d_{-j}$,

Si j gagne en déclarant $d_j = \langle s, d \rangle$,
alors,
j gagne aussi déclarant $d'_j = \langle s', d' \rangle$ avec $s' \subseteq s$, $d' \geq d$.

Remarque : La fonction d'allocation pour les enchères de Vickrey est monotone

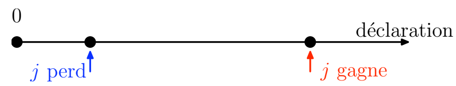


Enchère de Vickrey : la valeur critique

Theorem (Nisan02)

Soit A un algorithme d'allocation monotone. Pour n'importe d_{-j}, il existe une unique valeur critique $v_c(d_{-j}) \in R \cup \infty+$, tel que $\forall v_j > v_c(d_{-j})$, j gagne sinon il perd.

Preuve :



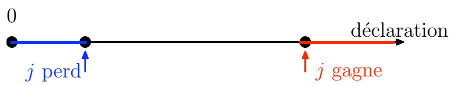
□

Enchère de Vickrey : la valeur critique

Theorem (Nisan02)

Soit A un algorithme d'allocation monotone. Pour n'importe d_{-j}, il existe une unique valeur critique $v_c(d_{-j}) \in R \cup \infty+$, tel que $\forall v_j > v_c(d_{-j})$, j gagne sinon il perd.

Preuve :



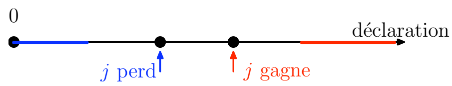
□

Enchère de Vickrey : la valeur critique

Theorem (Nisan02)

Soit A un algorithme d'allocation monotone. Pour n'importe d_{-j}, il existe une unique valeur critique $v_c(d_{-j}) \in R \cup \infty+$, tel que $\forall v_j > v_c(d_{-j})$, j gagne sinon il perd.

Preuve :



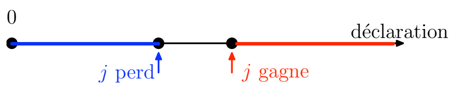
□

Enchère de Vickrey : la valeur critique

Theorem (Nisan02)

Soit A un algorithme d'allocation monotone. Pour n'importe d_{-j}, il existe une unique valeur critique $v_c(d_{-j}) \in R \cup \infty+$, tel que $\forall v_j > v_c(d_{-j})$, j gagne sinon il perd.

Preuve :



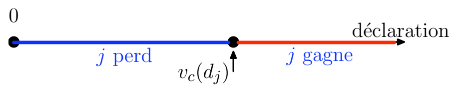
□

Enchère de Vickrey : la valeur critique

Theorem (Nisan02)

Soit A un algorithme d'allocation monotone. Pour n'importe d_{-j}, il existe une unique valeur critique $v_c(d_{-j}) \in R \cup \infty+$, tel que $\forall v_j > v_c(d_{-j})$, j gagne sinon il perd.

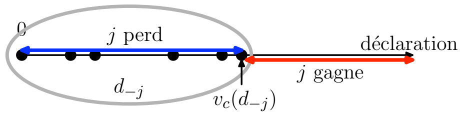
Preuve :



□

Enchère de Vickrey

- la valeur seuil $= \max\{d_j : j \neq \arg\max\{d_j\}\}$.



Definition
La fonction de paiement associé à un algorithme d'allocation monotone A est basée sur la valeur critique.

$$\begin{cases} p_j = v_c(d_{-j}) & \textit{si} \quad j \textit{ gagne} \\ p_j = 0 & \textit{sinon} \quad (j \textit{ perd}) \end{cases}$$

Mécanisme vérace.

Théorème
Tout mécanisme tel que

1. la fonction d'allocation A est monotone
2. la fonction de paiement A est basée sur la valeur critique.

est vérace.

Théorème
Les mécanisme VCG est un mécanisme vérace.

Allocation de ressources pour une arête.

Analogie avec le problème du sac-à-dos :

Données :
- un sac-à-dos de capacité b
- n objets de poids $s_1, \ldots, s_n$ et de valeurs $d_1, \ldots, d_n$

Question : Calculer un ensemble $S \subseteq \{1, \ldots, n\}$ tel que $\sum_{i \in S} d_i$ soit maximum tel que $\sum_{i \in S} s_i \leq b$

1/2-approximation [Mu'alem et Nisan 2002]

Considérons l'algorithme suivant [Johnson1973] $MAX(A_1, A_2)$:

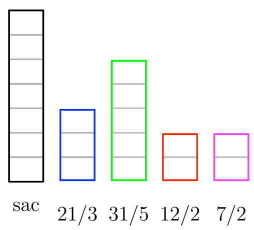
- Soit A_1 : l'algorithme glouton utilisant l'ordre suivant $\frac{d_i}{|s_i|}$
- Soit A_2 : l'algorithme glouton utilisant l'ordre suivant d_i
- retourne la solution qui est la meilleur de des deux. Exemple

Théorème : L'algorithme d'allocation $MAX(A_1, A_2)$ est une 1/2-approximation

Comment construire un mécanisme vérace sachant que cet algorithme sera la fonction d'allocation ?

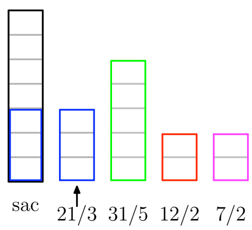
Suite

Exemple de l'algorithme glouton avec l'ordre $\frac{d_i}{|s_i|}$



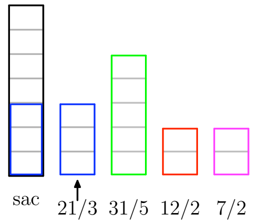
Retour

Exemple de l'algorithme glouton avec l'ordre $\frac{d_i}{|s_i|}$



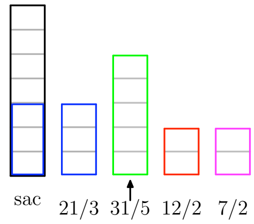
Retour

Exemple de l'algorithme glouton avec l'ordre $\frac{d_i}{|s_i|}$



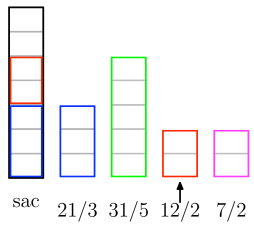
Retour

Exemple de l'algorithme glouton avec l'ordre $\frac{d_i}{|s_i|}$



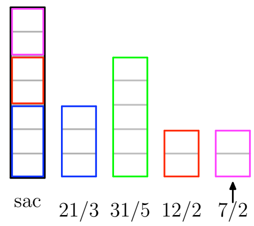
Retour

Exemple de l'algorithme glouton avec l'ordre $\frac{d_i}{|s_i|}$



Retour

Exemple de l'algorithme glouton avec l'ordre $\frac{d_i}{|s_i|}$



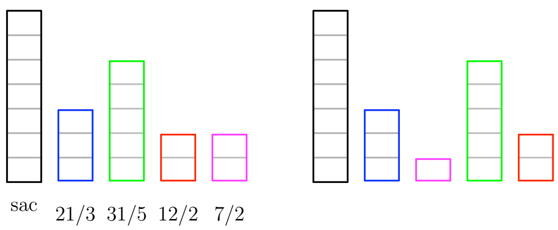
Retour

Propriété de l'algorithme glouton A_i utilisant l'ordre r

Algorithme A_i monotone ? oui car

- Considérons la configuration où j gagne en déclarant $\langle s_j : d_j \rangle$
- Gagne-t-il en déclarant $\langle s'_j : d'_j \rangle$ avec $s'_j \leq s_j$ et $d'_j \leq d_j$?

$$r(\langle s'_j : d'_j \rangle) \leq r(\langle s_j : d_j \rangle)$$

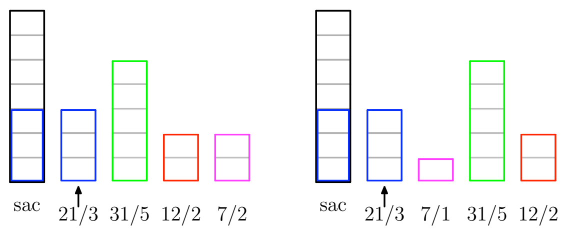


Propriété de l'algorithme glouton A_i utilisant l'ordre r

Algorithme A_i monotone ? oui car

- Considérons la configuration où j gagne en déclarant $\langle s_j : d_j \rangle$
- Gagne-t-il en déclarant $\langle s'_j : d'_j \rangle$ avec $s'_j \leq s_j$ et $d'_j \leq d_j$?

$$r(\langle s'_j : d'_j \rangle) \leq r(\langle s_j : d_j \rangle)$$

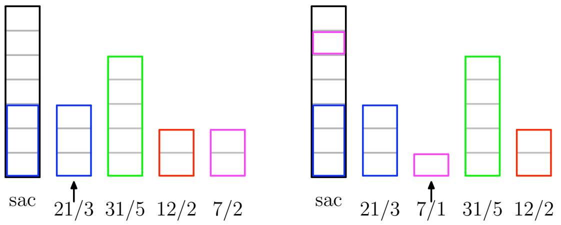


Propriété de l'algorithme glouton A_i utilisant l'ordre r

Algorithme A_i monotone ? oui car

- Considérons la configuration où j gagne en déclarant $\langle s_j : d_j \rangle$
- Gagne-t-il en déclarant $\langle s'_j : d'_j \rangle$ avec $s'_j \leq s_j$ et $d'_j \leq d_j$?

$$r(\langle s'_j : d'_j \rangle) \leq r(\langle s_j : d_j \rangle)$$

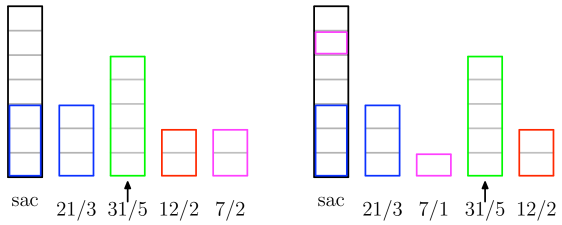


Propriété de l'algorithme glouton A_i utilisant l'ordre r

Algorithme A_i monotone ? oui car

- Considérons la configuration où j gagne en déclarant $\langle s_j : d_j \rangle$
- Gagne-t-il en déclarant $\langle s'_j : d'_j \rangle$ avec $s'_j \leq s_j$ et $d'_j \leq d_j$?

$$r(\langle s'_j : d'_j \rangle) \leq r(\langle s_j : d_j \rangle)$$

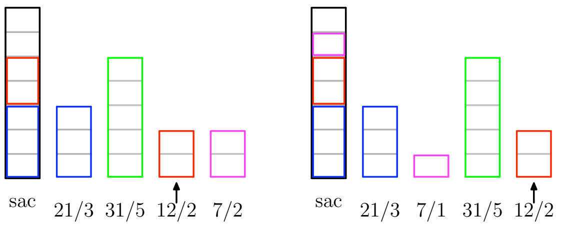


Propriété de l'algorithme glouton A_i utilisant l'ordre r

Algorithme A_i monotone ? oui car

- Considérons la configuration où j gagne en déclarant $\langle s_j : d_j \rangle$
- Gagne-t-il en déclarant $\langle s'_j : d'_j \rangle$ avec $s'_j \leq s_j$ et $d'_j \leq d_j$?

$$r(\langle s'_j : d'_j \rangle) \leq r(\langle s_j : d_j \rangle)$$

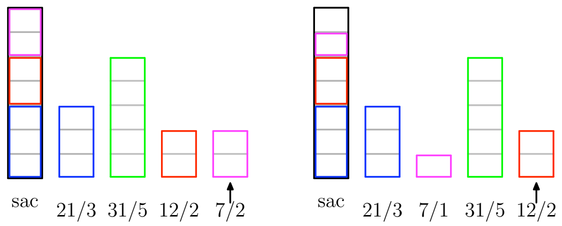


Propriété de l'algorithme glouton A_i utilisant l'ordre r

Algorithme A_i monotone ? oui car

- Considérons la configuration où j gagne en déclarant $\langle s_j : d_j \rangle$
- Gagne-t-il en déclarant $\langle s'_j : d'_j \rangle$ avec $s'_j \leq s_j$ et $d'_j \leq d_j$?

$$r(\langle s'_j : d'_j \rangle) \leq r(\langle s_j : d_j \rangle)$$



Fonction d'allocation monotone

$$MAX(A_1, A_2) = \max \begin{pmatrix} A_1 & \text{algo. glouton utilisant l'ordre } \frac{d_i}{|S_i|}, \\ A_2 & \text{algo. glouton utilisant l'ordre } d_i \end{pmatrix}.$$

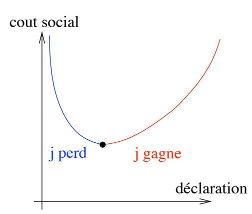
Remarque

Le maximun de deux ou plus fonctions d'allocation monotone n'est pas monotone.

Fonction allocation monotone *bitonique*

Définition :

La fonction d'allocation monotone est bitonique par rapport à son coût social si pour d_{-j} :

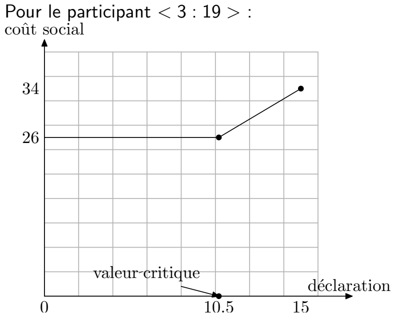


Lemme [Mu'alem et Nisan 2002] :

Un algorithme qui consiste à sélectionner le maximum de 2 ou + d'algorithme bitonique est un algorithme monotone.

l'algorithme glouton A_i utilisant l'ordre r est bitonique

- Configuration : 7 exemplaires d'un $\hat{m}$ objet et déclarations $= (< 1 : 7 >, < 3 : 19 >, < 2 : 12 >, < 2 : 7 >)$,
- liste des gagnants $= (< 1 : 7 >, < 3 : 19 >, < 2 : 12 >)$,
- Pour le participant $< 3 : 19 >$:



1/2-approximation

Considérons l'algorithme suivant $MAX(A_1, A_2)$:

- Soit A_1 : l'algorithme glouton utilisant l'ordre suivant $\frac{d_i}{|S_i|}$
- Soit A_2 : l'algorithme glouton utilisant l'ordre suivant d_i
- retourne la solution qui est la meilleur de des deux.

Théorème : Le mécanisme

- ayant $MAX(A_1, A_2)$ comme sa fonction d'allocation
- ayant sa fonction de paiement basée sur la valeur critique de $MAX(A_1, A_2)$

est vérace.

En résumé

	1 arête	Chemin	Arbre	Graphe
requête unitaire	Poly mécanisme VCG	Poly mécanisme VCG	en cours 1/4-approx.	NP-Complet $O(1/\sqrt{n})$-approx [2002]
requête non-unitaire	1/2-approx FPTAS [2005]	1/3-approx [2006]		$O(1/\sqrt{n})$-approx [2002]

Problème du plus court chemin

Données : un graphe $G = (V, E)$

- Un sommet w est un agent w
- Chaque sommet w a un type t_w (un poids)
- deux sommets de x et y distincts

Objectif global :

Trouver un chemin C de + court chemin entre x et y.

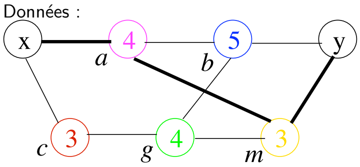
Objectif local de w :

minimiser le coût de w avec

$$cout(w) = \begin{cases} 0 & \text{si } C \text{ ne traverse pas } w \\ t_w & \text{si } C \text{ traverse } w \end{cases}$$

C'est-à-dire

Données :



Objectif global : le $\oplus$ court chemin $C = \{x, a, m, y\}$ de poids 7

Objectif local :

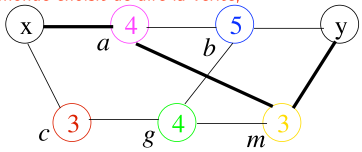
- $cout(b) = cout(c) = cout(g) = 0$
- $cout(a) = 4$ et $cout(m) = 3$

Le mécanisme sur le plus court chemin.

- $A(\overrightarrow{d})$: algo. du $\oplus$ court chemin avec $\overrightarrow{d}$ comme poids.
- Fixons i et d_{-i}. $L_{d_i=x}$ = la longueur du pcc si $d_i = x$.
- Posons $C_i = -L_{d_i=0} + L_{d_i=+\infty}$ et
- définission la prime
  $$p_i = \begin{cases} C_i & \text{si i est dans le plus court chemin} \\ 0 & \text{sinon} \end{cases}$$
- Chaque agent veut maximiser $u_i = p_i + v_i$.

Illustration : max $u_i = p_i + v_i$

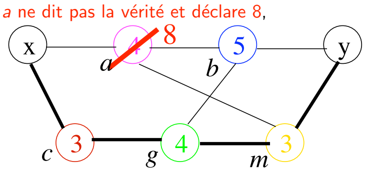
Si tout le monde choisit de dire la vérité,



- pour les agents c, g, m ($\notin pcc$) : $p_i = 0$ et $u_i = 0$
- pour l'agent a ($\in pcc$) : $p_a = 7$ et $u_a = 3$
  - $p_a = -L_{d_a=0} + L_{d_a=+\infty}$
  - $L_{d_a=0} = 3$ et $L_{d_a=+\infty} = d_{c-g-m} = 10$
  - $v_a = -t_a = -4$ et $u_a = 3 \geq 0$
- pour l'agent m ($\in pcc$) : $p_m = 5$ et $u_m = 2$
  - $L_{d_m=0} = d_a = 4$ et $L_{d_m=+\infty} = d_{a-b} = 9$
  - $p_m = -4 + 9 = 5$ et $u_m = -t_m + 5 \geq -3 + 5 \geq 0$

Illustration : max $u_i = p_i + v_i$

Si l'agent a ne dit pas la vérité et déclare 8,

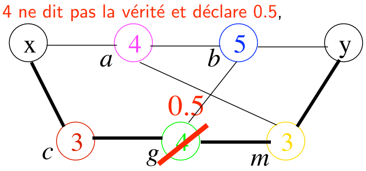


- pour l'agent a ($\notin pcc$) : $p_a = 0$ et $u_a = 0$.

  L'agent a ne tire pas profit de son mensonge

Illustration : max $u_i = p_i + v_i$

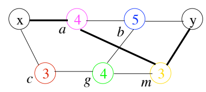
Si l'agent 4 ne dit pas la vérité et déclare 0.5,

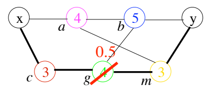


- pour l'agent 4 ($\in pcc$) : $p_4 = 1$ et $u_4 = -3$.
  - $p_4 = -L_{d_4=0} + L_{d_4=+\infty} = -6 + 7 = 1$
  - $u_4 = -4 + 1 \leq 0$

L'agent 4 ne tire pas profit de son mensonge

Signification de la stratégie dominante





Stratégie "dire la verité" "déclarer 0.5 au lieu de 2"

- stratégie dominante s_i : $\forall d_{-i}$, $\forall d_i$, $u_i(s_i, d_{-i}) \geq u_i(d_i, d_{-i})$
- stratégie que 4 a d'annoncer 0.5 n'est pas dominante car

$$\underbrace{u_i(s_i, d_{-i})}_{-4+1} \leq \underbrace{u_i(t_i, d_{-i})}_{0}$$

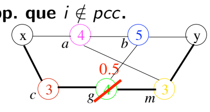
mécanisme vérace ?

Lemme 1 : $C_i = -L_{d_i=0} + L_{d_i=+\infty}$ est une valeur seuil :

1. $d_i < C_i \Rightarrow$ l'agent i est dans le plus court chemin.
2. $d_i > C_i \Rightarrow$ l'agent i n'est pas dans le plus court chemin.
3. $t_i = C_i \Rightarrow \forall \vec{d}, u_i(\vec{d}) = 0$.

Preuve **du point 1 : par contradiction : supp. que** $i \notin pcc$.

- **longueur du** pcc **=** $L_{d_i=+\infty}$
- **Donc** $L_{d_i=+\infty} \leq L_{d_i=0} + d_i$
- **Donc** $C_i \leq d_i$

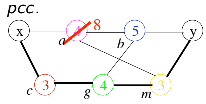


$d_i = 0.5, C_i = -6 + 7$

Preuve du Lemme.

du point 2 : par contradiction : supp. que $i \in pcc$.

- longueur du pcc = $d_i + L_{d_i=0}$
- Donc $d_i + L_{d_i=0} \leq L_{d_i=+\infty}$
- Donc $d_i \leq C_i$



$d_i = 8, C_i = -5 + 10$

du point 3 :

- Si $i \notin pcc$, alors $u_i = 0$
- Si $i \in pcc$, $u_i = -t_i + C_i = 0$.

mécanisme vérace ?

Proposition 1 : le mécanisme est vérace.
Preuve :

- $u_i(x, d_{-i})$ = utilité de l'agent i quand i annonce x.
- Si $t_i \leq C_i$. $\forall d_i$, $u_i(t_i) \geq u_i(d_i)$
  - i déclare t_i : $i \in pcc$ et $u_i(t_i) = v_i + p_i = -t_i + C_i \geq 0$.
  - i déclare d_i : $u_i(t_i) \geq u_i(d_i)$
    - $d_i \leq C_i$: $i \in pcc$ et $u_i(d_i) = -t_i + C_i = u_i(t_i)$
      $i \in pcc$ dans les deux cas (le gain = la prime)
    - $d_i > C_i$: $i \notin pcc$ et $u_i(d_i) = 0$
      $u_i(t_i) \geq u_i(d_i)$

Preuve de la Proposition

- Si $t_i > C_i$: $\forall d_i$, $u_i(t_i) \geq u_i(d_i)$
  - i déclare t_i : $u_i(t_i) = 0$ car $i \notin pcc$.
  - i déclare d_i : $u_i(t_i) \geq u_i(d_i)$
    - $d_i \leq C_i$; $i \in pcc$. Donc $u_i(d_i) = -t_i + C_i \leq 0 = u_i(t_i)$
    - $d_i > C_i$; $i \notin pcc$ et $0 = u_i(t_i) = u_i(d_i)$

Conclusion

- Réservation de chemins avec mécanismes véraces :
  problème d'allocation + problème de paiement
- Problème d'allocation = problème du sac-à-dos (algorithmes d'approximation ayant de "bonnes" des propriétés)
- L'objectif ici est d'inciter les participants à dévoiler leurs besoins/valuation

Mais pas de maximiser le revenu de l'administrateur ! ! ! !

Question ?

Patrick Briest and Piotr Krysta and Berthold Vocking
Approximation techniques for utilitarian mechanism design.
STOC 2005.

Chandra Chekuri, Marcelo Mydlarz, F. Bruce Shepherd
Multicommodity Demand Flow in a Tree.
ICALP 2003.

Magnús M. Halldórsson
Approximations of Weighted Independent Set and Hereditary Subset Problems.
COCOON 99

Daniel Lehmann and Liadan Ita O'allaghan and Yoav Shoham
Truth revelation in approximately efficient combinatorial auctions A
Journal of ACM 2002.