À la recherche d'une solution "stable" : Deux problèmes mélant RO et Théorie des jeux

Eric Angel, Evripidis Bampis, Lélia Blin, Laurent Gourvès, Fanny Pascual

LaMI, Université d'Évry

Stabilité

- Jeux coopératifs

  un ensemble V d'agents

  une fonction caractéristique f qui à tout $S \subseteq V$ associe une valeur $f(S)$

  que les agents se partagent s'ils coopèrent

  Coeur de f : $x \in \mathbb{R}^V$ t.q. $\sum_{i \in V} x_i = f(V)$ et $\forall S \subset V : \sum_{i \in S} x_i \geq f(S)$

  Stabilité=Coeur

- Jeux non-coopératifs

  un ensemble V d'agents

  un ensemble de stratégies pour chaque agent

  une fonction f qui selon la stratégie choisie par chaque agent associe un vecteur $x \in \mathbb{R}^V$

  Équilibre de Nash : Situation dans laquelle aucun agent i ne peut améliorer x_i en modifiant unilatéralement sa stratégie

  Stabilité=équilibre de Nash

Plan

1. Allocation de coût équitable pour le jeu de l'arbre couvrant de poids minimal

   Eric Angel, Evripidis Bampis, Lélia Blin, Laurent Gourvès

2. Ordonnancement de tâches sur deux machines et stabilité approchée

   Eric Angel, Evripidis Bampis, Fanny Pascual

Allocation de coût équitable pour le jeu de l'arbre couvrant de poids minimal.

Jeu de l'ACPM

- Données :
  - Un graphe $G = (V, E)$ connexe dont les arêtes sont valuées (fonction coût c)
  - Un sommet racine $r \in V$ qui doit délivrer une information à travers G aux agents (sommets de $V \backslash r$)

Jeu de l'ACPM

- Données :
  - Un graphe $G = (V, E)$ connexe dont les arêtes sont valuées (fonction coût c)
  - Un sommet racine $r \in V$ qui doit délivrer une information à travers G aux agents (sommets de $V \backslash r$)
- Problème :

1. Déterminer le sous-graphe de diffusion de l'information
2. Répercuter le coût du sous-graphe aux agents

Jeu de l'ACPM

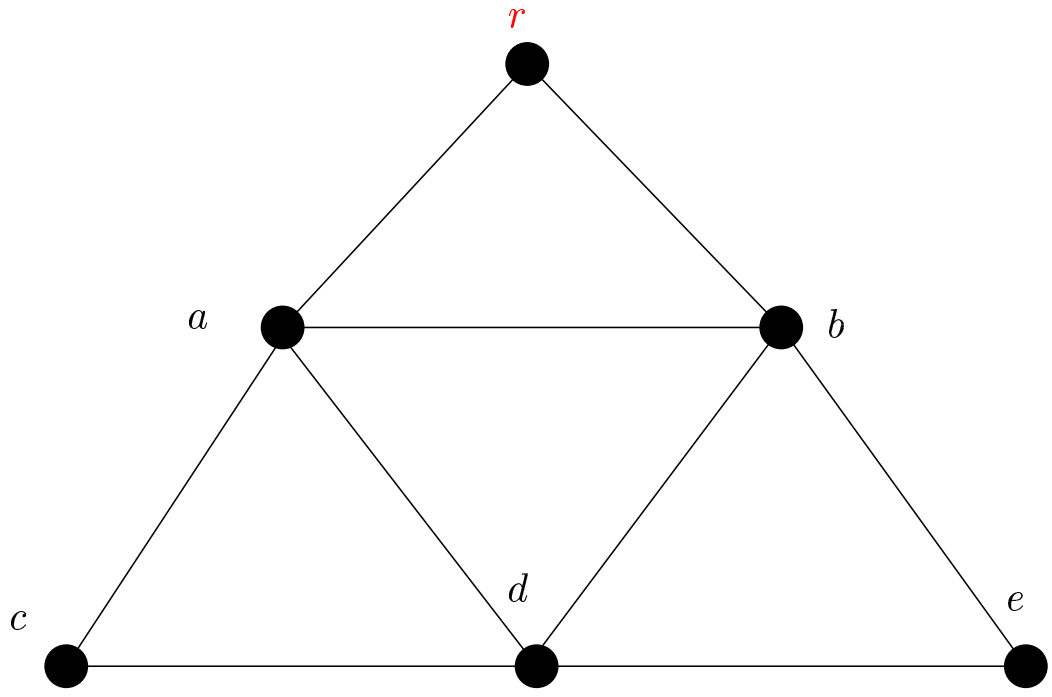
- Données :
  - Un graphe $G = (V, E)$ connexe dont les arêtes sont valuées (fonction coût c)
  - Un sommet racine $r \in V$ qui doit délivrer une information à travers G aux agents (sommets de $V \backslash r$)
- Problème :

1. Déterminer le sous-graphe de diffusion de l'information $\Rightarrow$ ACPM
2. Répercuter le coût du sous-graphe aux agents
   $\Rightarrow$ Allocation équitable

Stabilité : Allocation dans le coeur du jeu coopératif

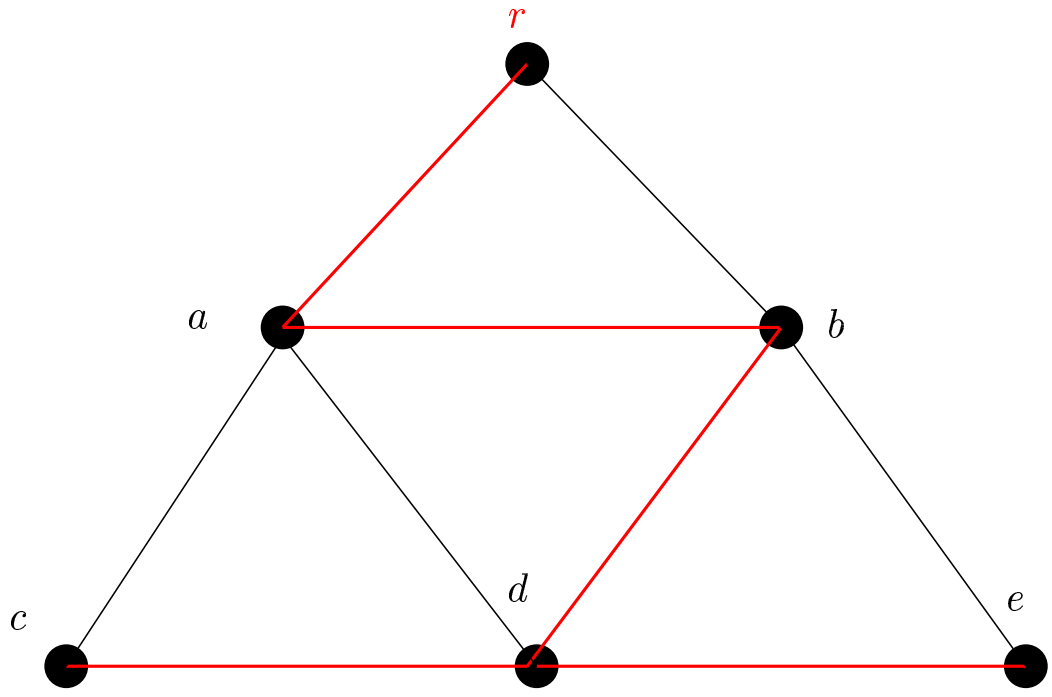
Allocation de Bird

Le prix demandé à un agent est égal au coût de la première arête dans l'unique chemin qui le sépare de la racine



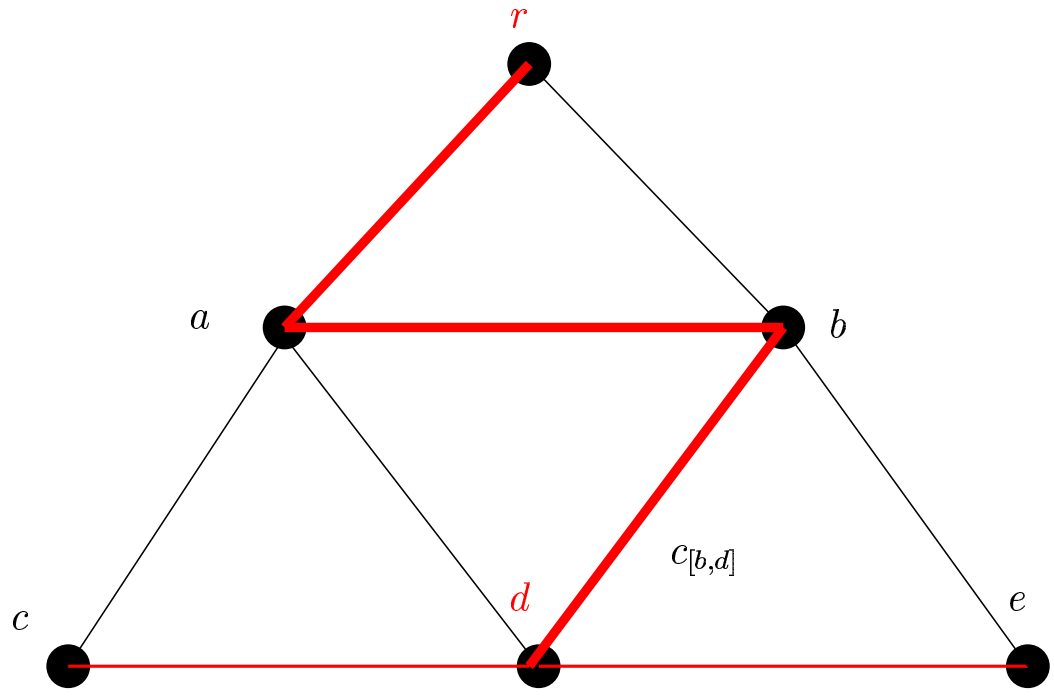
Allocation de Bird

Le prix demandé à un agent est égal au coût de la première arête dans l'unique chemin qui le sépare de la racine



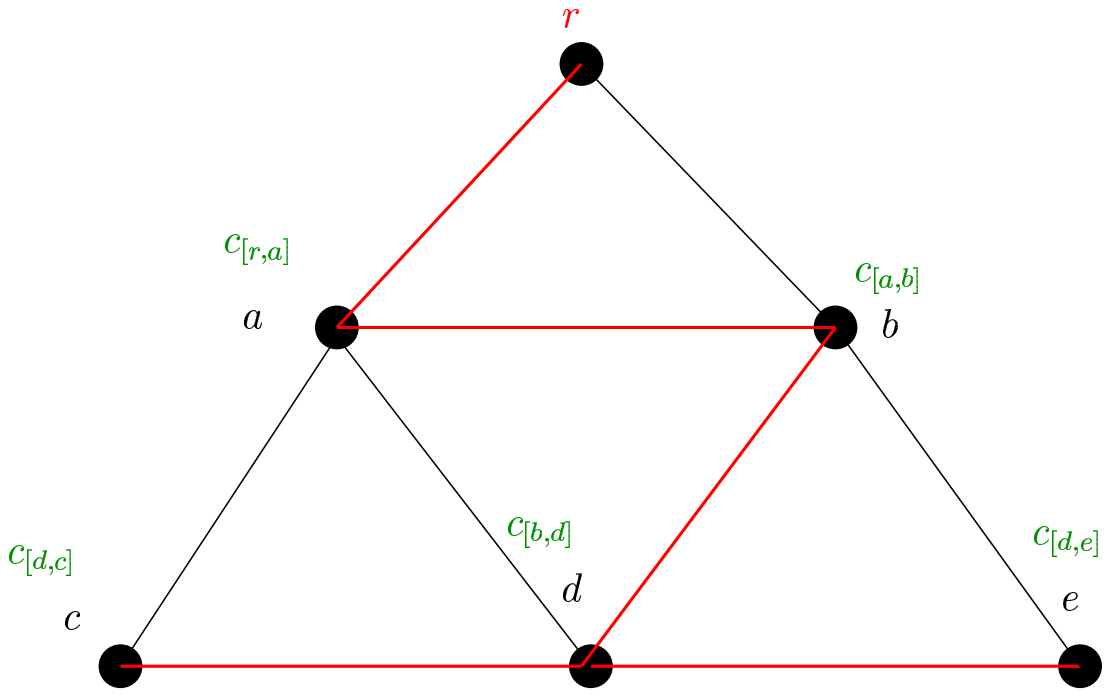
Allocation de Bird

Le prix demandé à un agent est égal au coût de la première arête dans l'unique chemin qui le sépare de la racine



Le prix demandé à d, noté $\beta(T, d)$, est $c_{[b,d]}$

Allocation de Bird

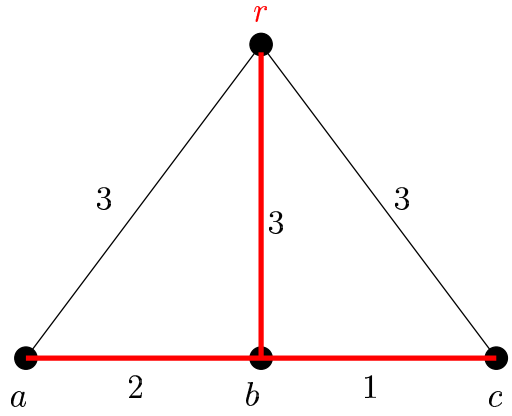


Allocation $(c_{[r,a]}, c_{[a,b]}, c_{[d,c]}, c_{[b,d]}, c_{[d,e]})$

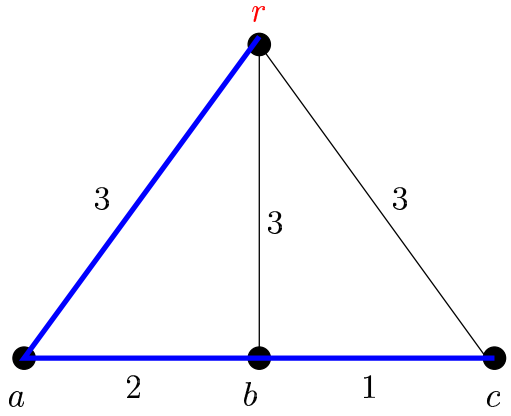
Toute allocation de Bird appartient au coeur

Mécontentement

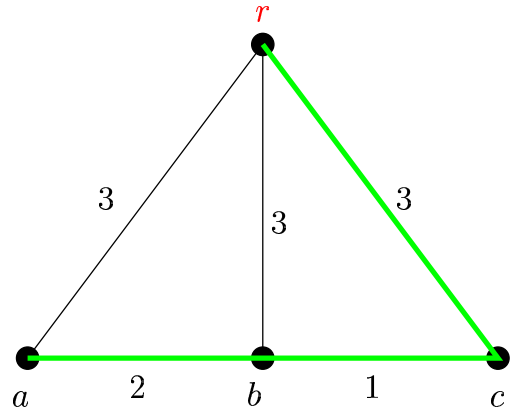
Plusieurs ACPM $\Rightarrow$ plusieurs allocations de Bird



$(2, 3, 1)$



$(3, 2, 1)$



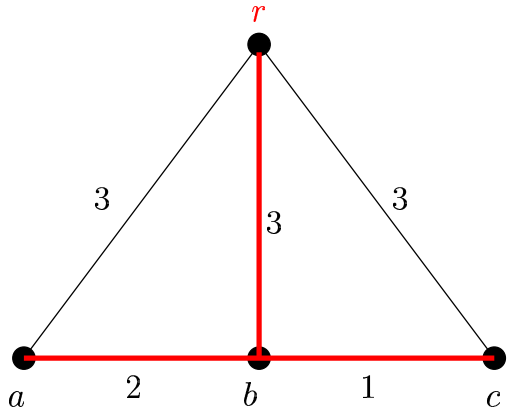
$(2, 1, 3)$

Du point de vue d'un agent, le prix demandé dépend grandement de l'ACPM

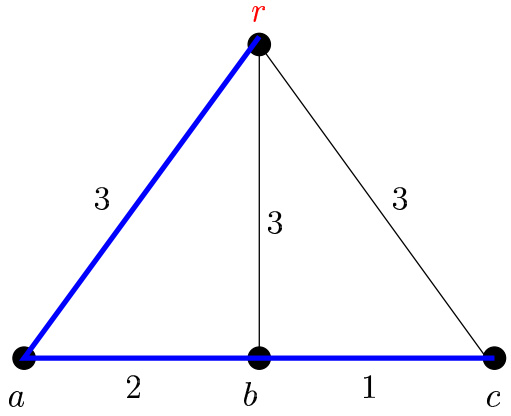
$F(v)$ = ensemble des prix pouvant être demandés à v
$F_{min}(v) = \min F(v)$

Mécontentement

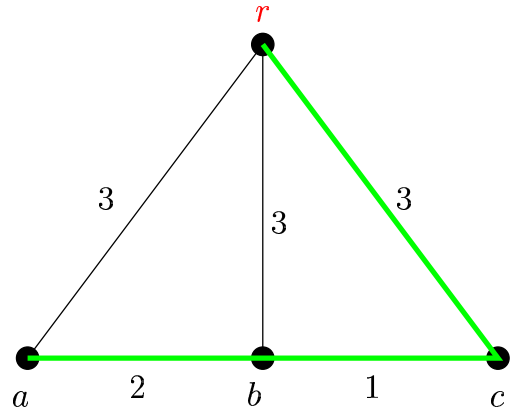
Plusieurs ACPM $\Rightarrow$ plusieurs allocations de Bird



$(2, 3, 1)$



$(3, 2, 1)$

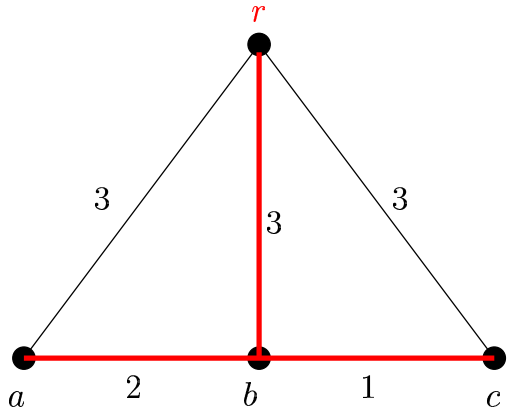


$(2, 1, 3)$

Mécontentement d'un agent = prix actuel / prix minimal

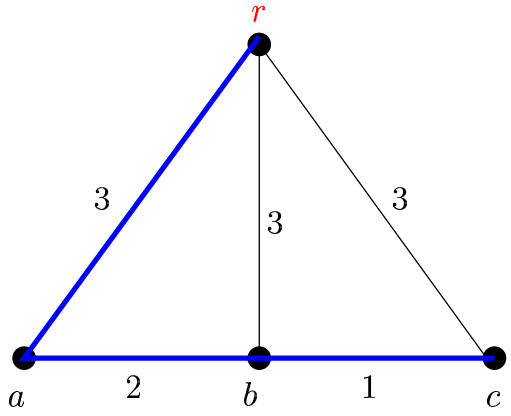
Mécontentement de l'agent v : $\Delta(T, v) = \frac{\beta(T,v)}{F_{min}(v)}$

Exemple



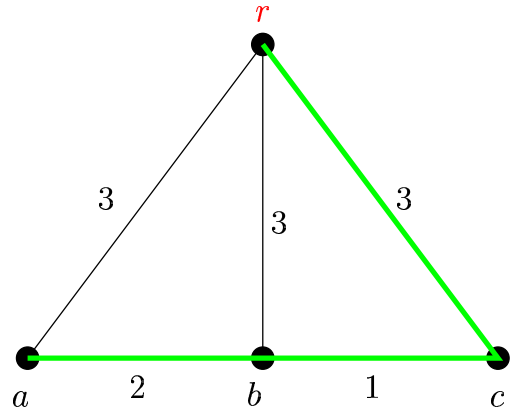
$(2, \mathbf{3}, 1)$

$\Delta(T, b) = 3$



$(3, \mathbf{2}, 1)$

$\Delta(T, b) = 2$



$(2, \mathbf{1}, 3)$

$\Delta(T, b) = 1$

Deux problèmes d'optimisation

1. SPANNING TREE-MWD : Déterminer un ACPM qui minimise le pire mécontentement

$$\min_{T} \max_{v \in V} \Delta(T, v)$$ sous contrainte que T est un ACPM

Deux problèmes d'optimisation

1. SPANNING TREE-MWD : Déterminer un ACPM qui minimise le pire mécontentement

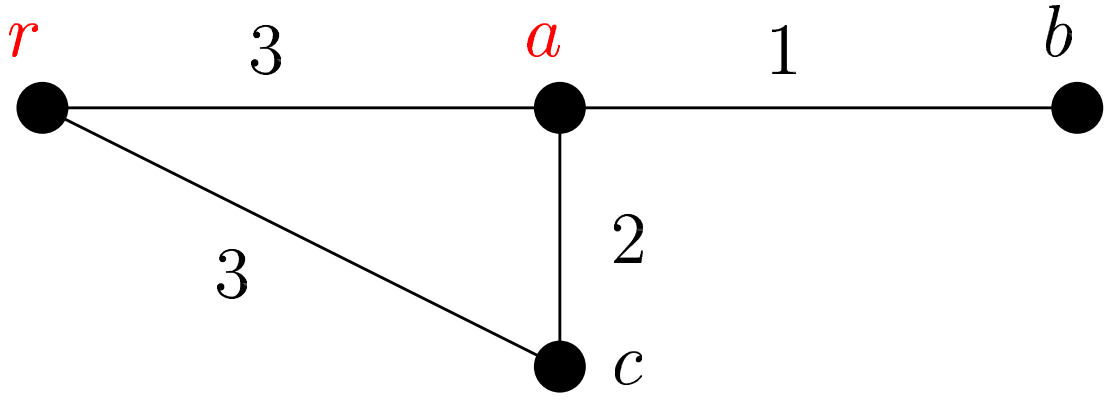
$$\min_{T} \max_{v \in V} \Delta(T, v) \text{ sous contrainte que } T \text{ est un ACPM}$$

2. SPANNING TREE-MAD : Déterminer un ACPM qui minimise le mécontentement moyen

$$\min_{T} \sum_{v \in V} \Delta(T, v) \text{ sous contrainte que } T \text{ est un ACPM}$$

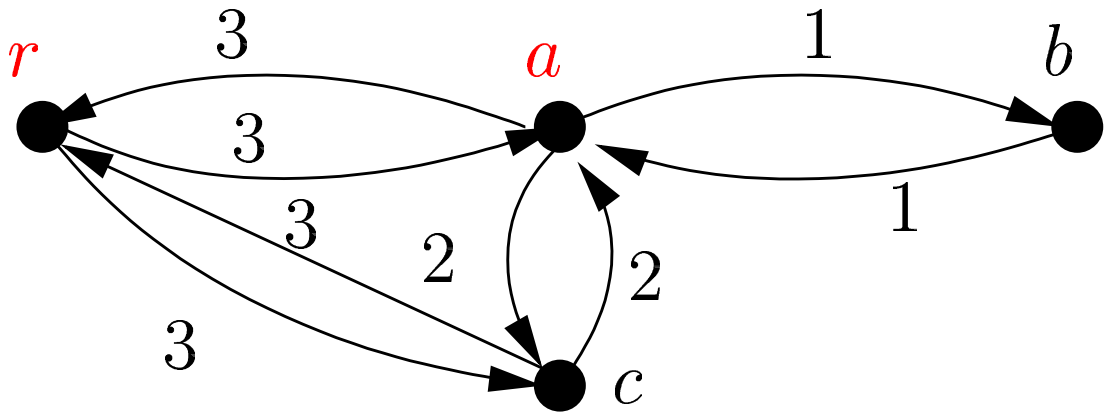
Connaître $F_{min}(v)$

Le prix demandé à un agent fait partie des poids des arêtes qui lui sont incidentes



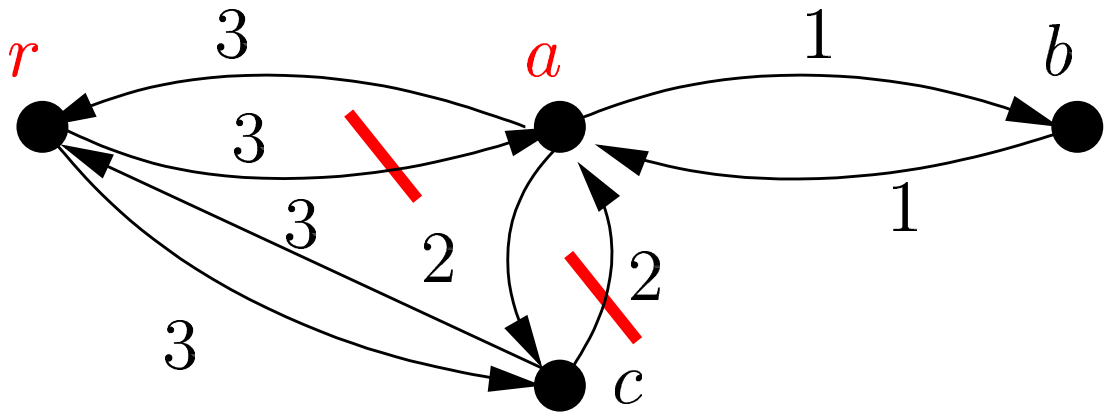
Mais le plus petit prix demandé n'est pas nécessairement la plus petite de ces valeurs

Connaître $F_{min}(v)$



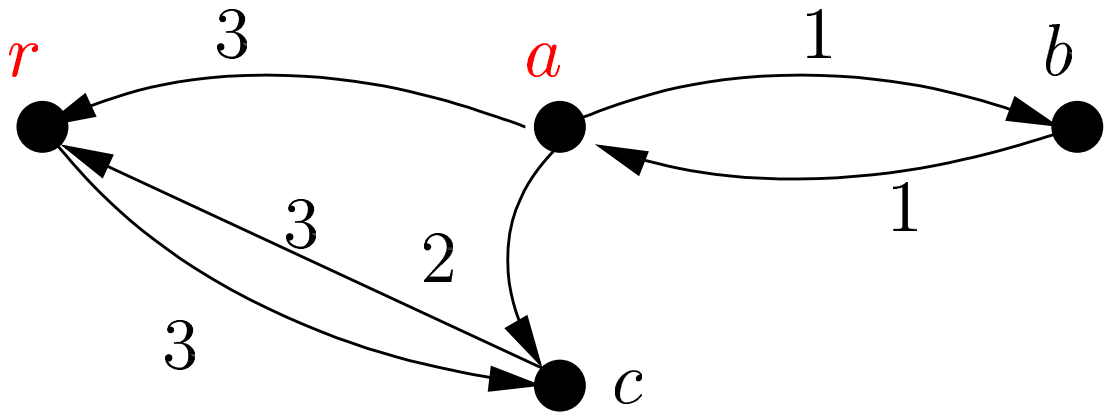
Besoin d'une orientation du graphe

Connaître $F_{min}(v)$



On force a à recevoir le prix 1

Connaître $F_{min}(v)$



Pas d'arborescence enracinée en r

Minimiser le pire mécontentement

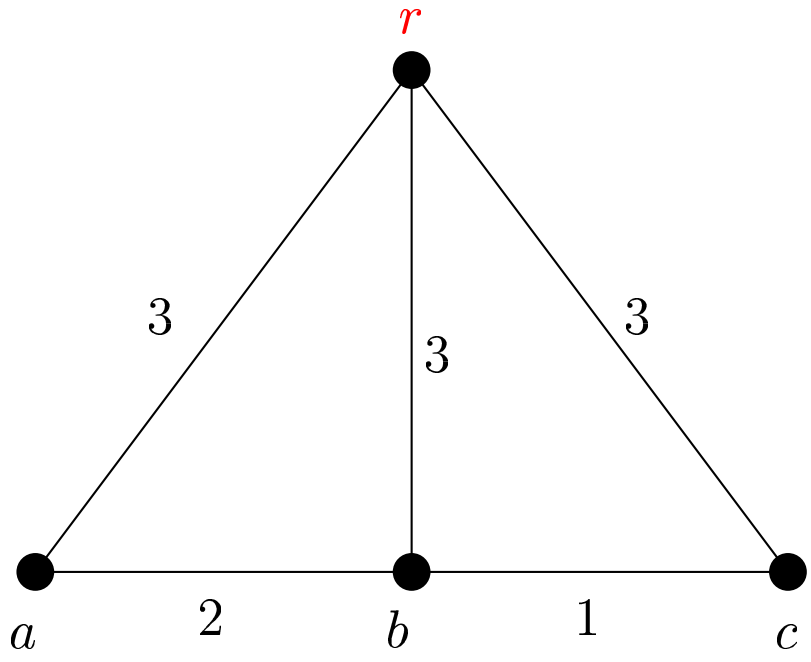
Idée de l'algorithme :

Déterminer un sous-graphe G' du graphe original G tel que tout ACPM de G' minimise le pire mécontentement

G' est obtenu en retirant successivement des arêtes de G

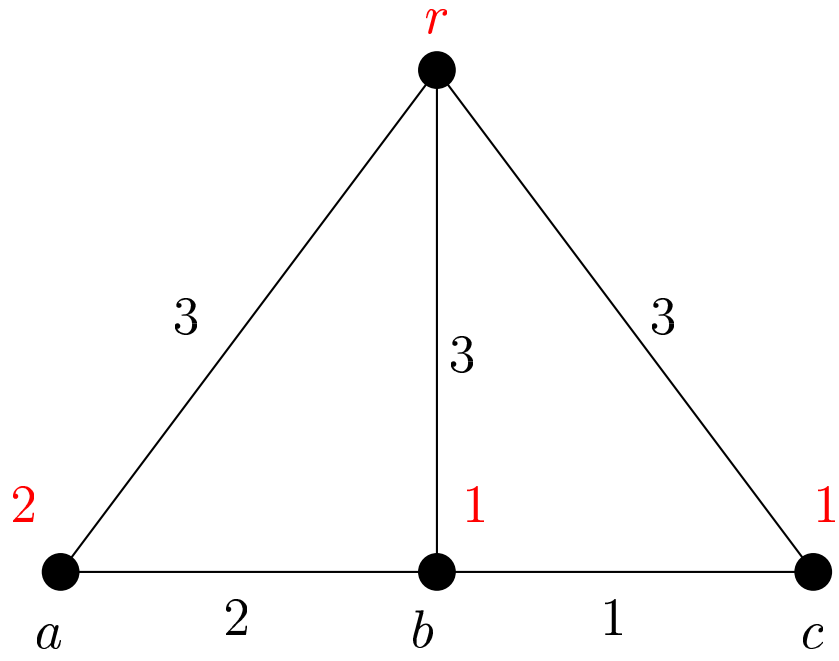
condition d'arrêt : G' ne contient plus d'ACPM

Exemple



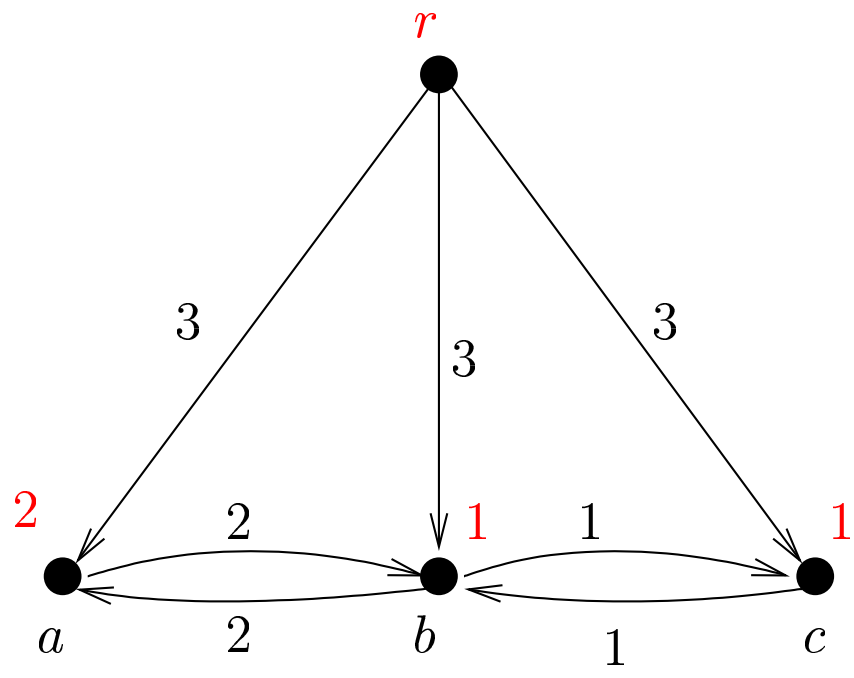
Tout ACPM a un poids $C_{opt} = 6$

Exemple



- Le plus petit prix reçu par a est 2 $\rightarrow F_{min}(a) = 2$
- Le plus petit prix reçu par b est 1 $\rightarrow F_{min}(b) = 1$
- Le plus petit prix reçu par c est 1 $\rightarrow F_{min}(c) = 1$

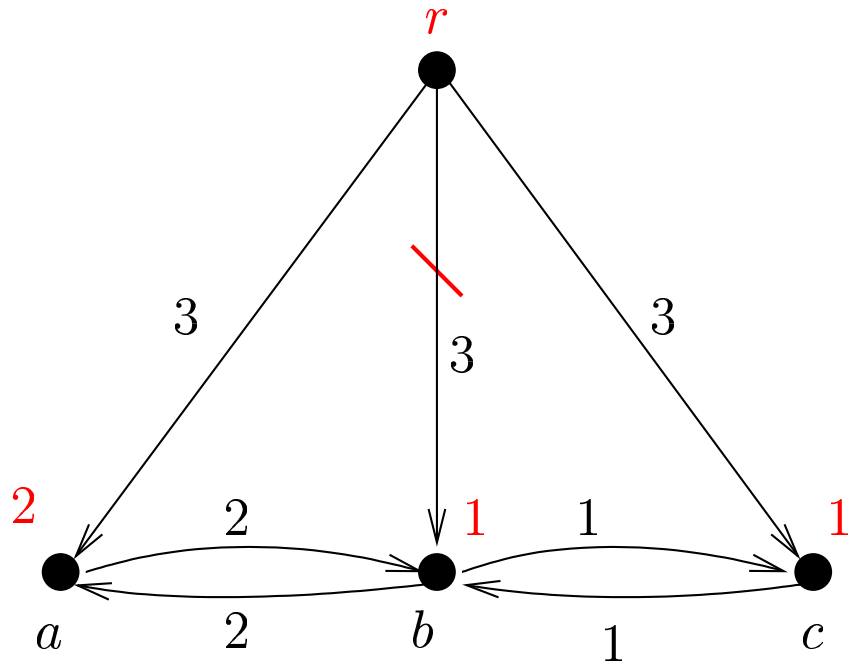
Exemple



Orientation du graphe

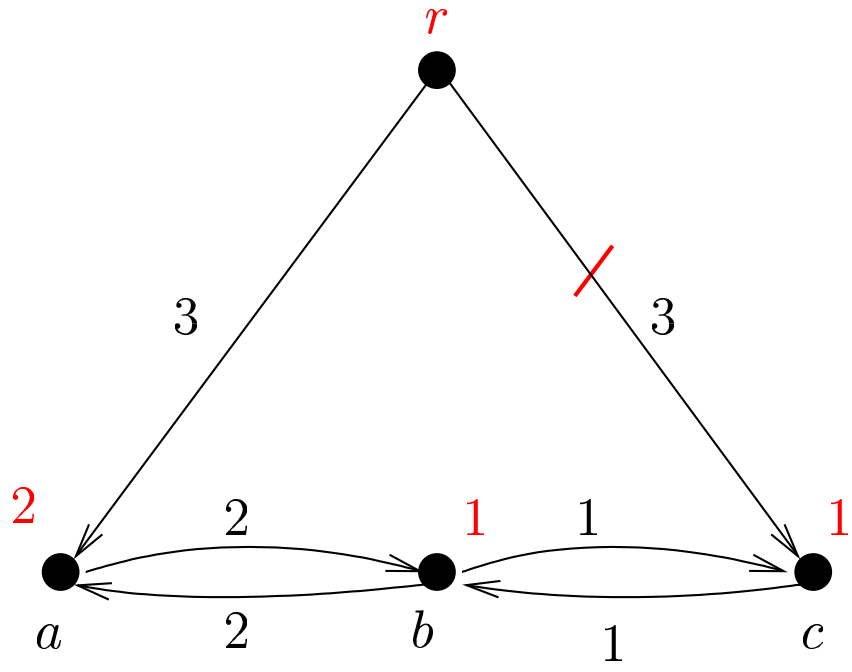
On retire successivement les arcs (x, y) tels que $c_{(x,y)}/F_{min}(y)$ est maximal

Exemple



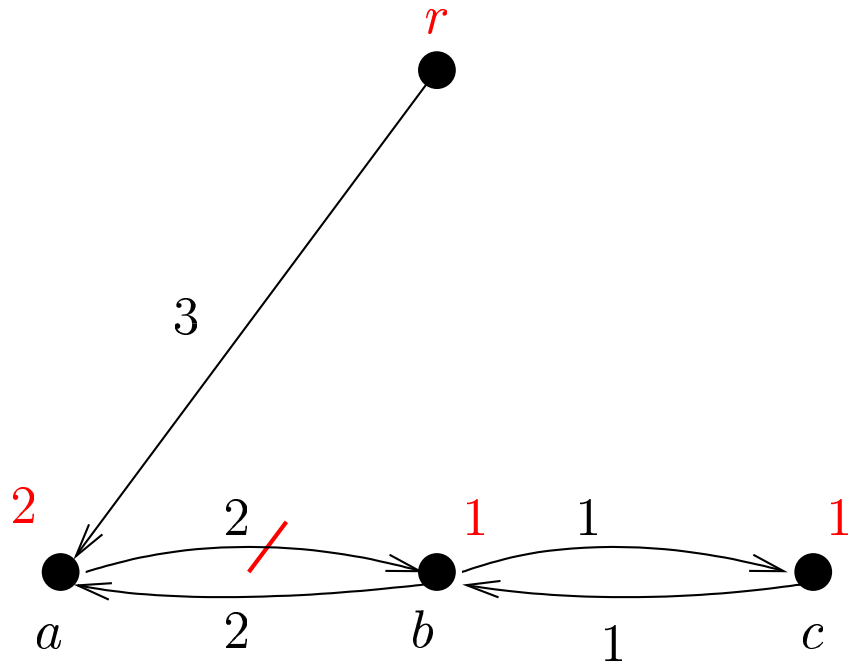
On effectue ce traitement tant qu'il existe une arborescence enracinée en r de poids C_{opt} qui couvre tous les sommets

Exemple



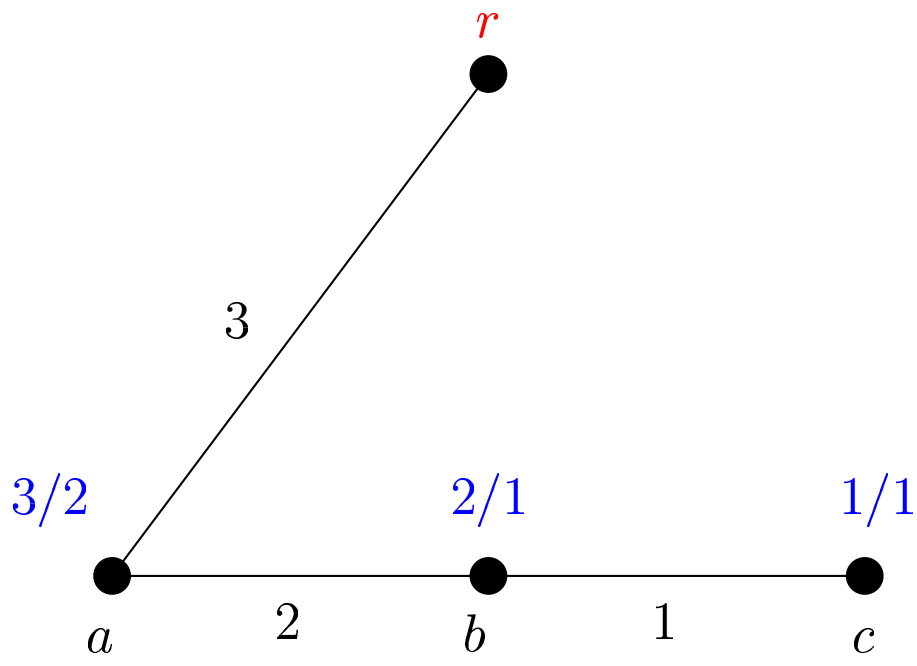
On effectue ce traitement tant qu'il existe une arborescence enracinée en r de poids C_{opt} qui couvre tous les sommets

Exemple



Stop : il n'existe plus d'arborescence enracinée en r qui couvre tous les sommets

Exemple



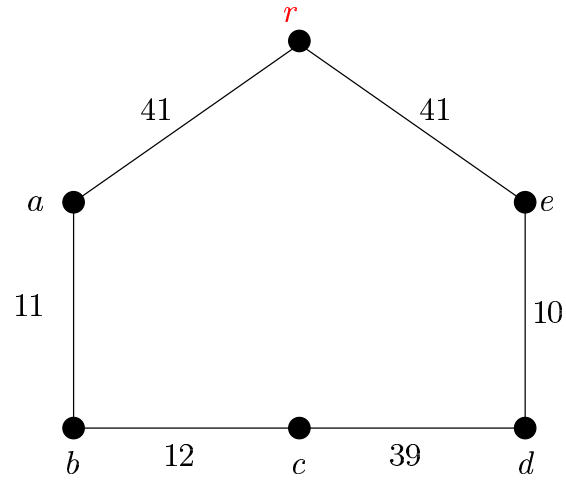
ACPM qui minimise le pire mécontentement

Mécontentement : Pire vs Moyen

On s'intéresse à la détermination d'un ACPM qui minimise le mécontentement moyen

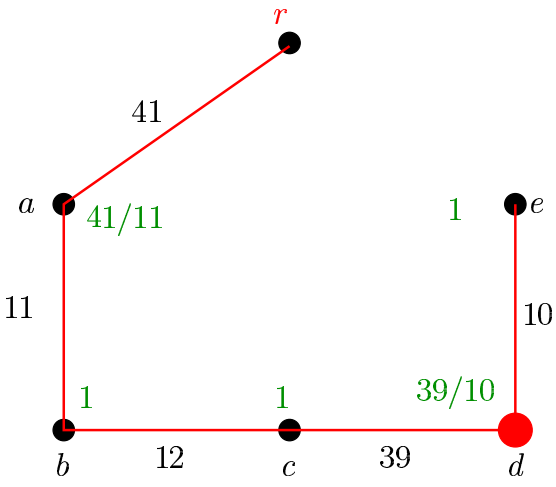
Remarque : Pas toujours concordance entre ACPM qui minimise le pire mécontentement et ACPM qui minimise le mécontentement moyen

Mécontentement : Pire vs Moyen

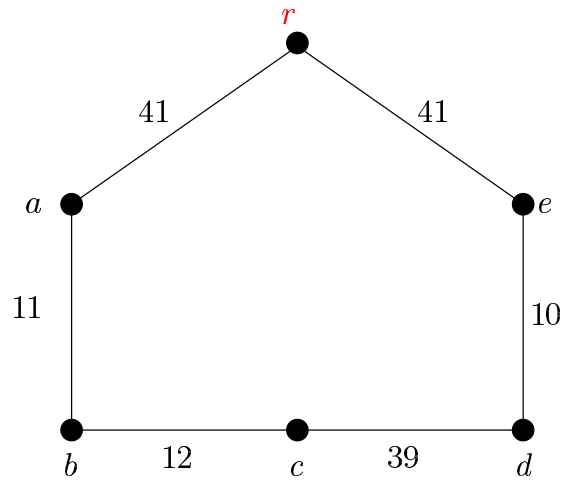


Instance

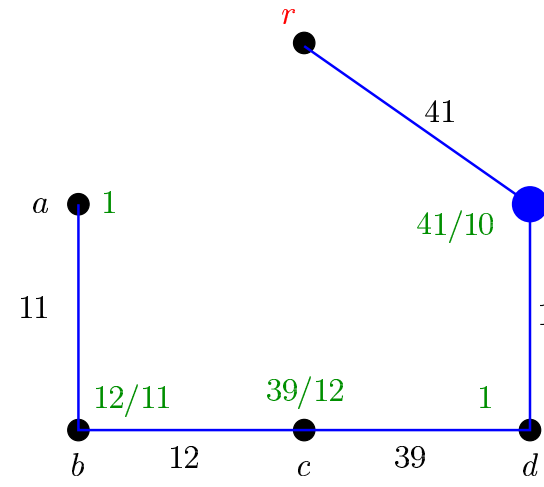
Mécontentement : Pire vs Moyen



Pire=3,9
Moyen= 2,12



Instance

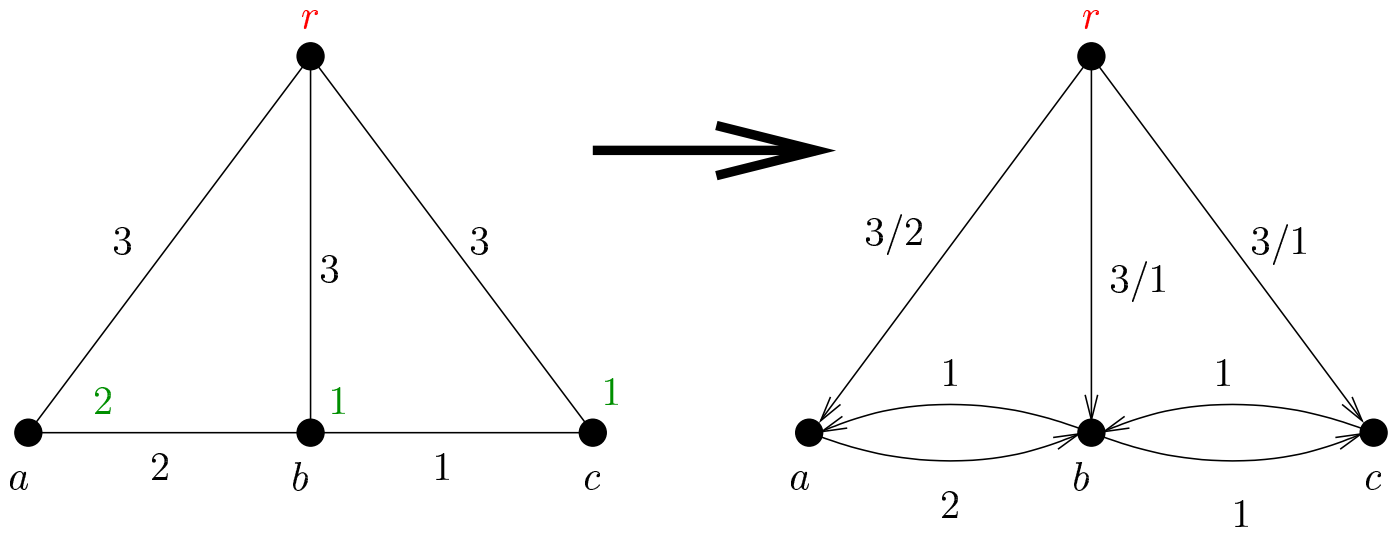


Pire=4,1
Moyen=2,08

Différence entre l'arbre qui minimise le pire mécontentement et celui qui minimise le mécontentement moyen

Minimiser le mécontentement moye

Idée : Remplacer le poids des arcs (u, v) par $\frac{c_{(u,v)}}{F_{min}(v)}$ puis déterminer une arborescence de poids minimal



Pb : On ne récupère pas toujours un ACPM

Minimiser le mécontentement moye

Deux critères à traiter de manière hiérarchique :

1. Le poids de l'arbre
2. Le mécontentement moyen

Minimiser le mécontentement moye

Deux critères à traiter de manière hiérarchique :

1. Le poids de l'arbre
2. Le mécontentement moyen

Du point de vue d'un arc (u, v) :

1. $c_{(u,v)}$
2. $c_{(u,v)}/F_{min}(v)$

Minimiser le mécontentement moye

Utilisation d'un coût composite $\tilde{c}$, combinaison des deux critères :

$$\tilde{c}_{(u,v)} = \lambda c_{(u,v)} + (1 - \lambda)\frac{c_{(u,v)}}{F_{min}(v)}$$

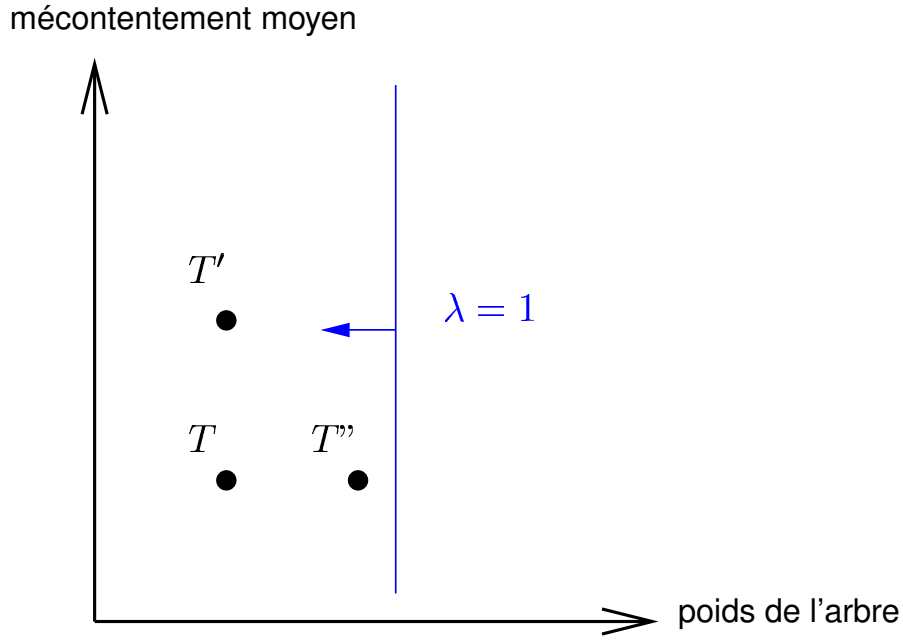
Déterminer un λ tel que tout ACPM vis à vis de $\tilde{c}$ soit l'optimum recherché

Minimiser le mécontentement moye

Utilisation d'un coût composite $\tilde{c}$, combinaison des deux critères :

$$\tilde{c}_{(u,v)} = \lambda c_{(u,v)} + (1 - \lambda)\frac{c_{(u,v)}}{F_{min}(v)}$$

Déterminer un λ tel que tout ACPM vis à vis de $\tilde{c}$ soit l'optimum recherché

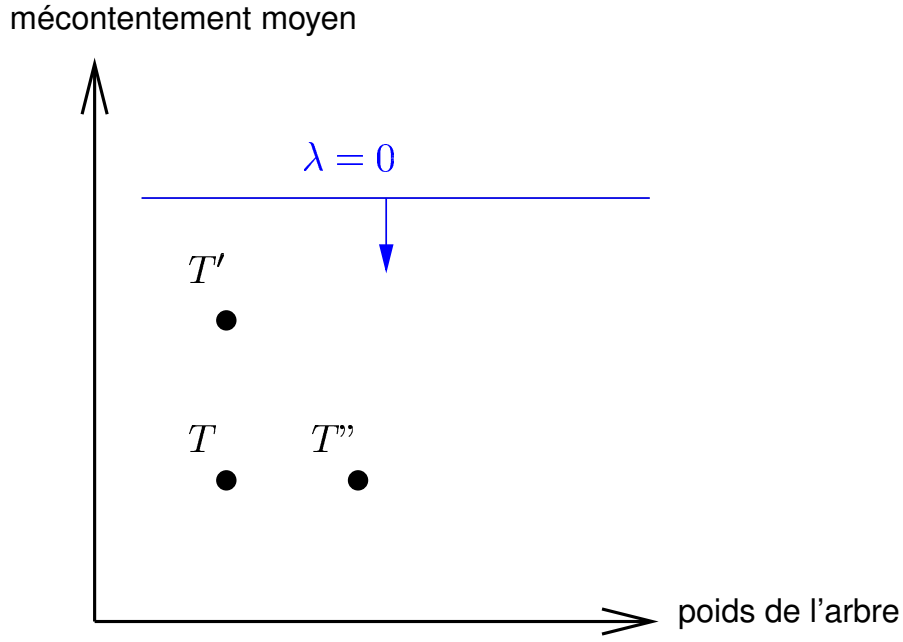


Minimiser le mécontentement moye

Utilisation d'un coût composite $\tilde{c}$, combinaison des deux critères :

$$\tilde{c}_{(u,v)} = \lambda c_{(u,v)} + (1 - \lambda)\frac{c_{(u,v)}}{F_{min}(v)}$$

Déterminer un λ tel que tout ACPM vis à vis de $\tilde{c}$ soit l'optimum recherché

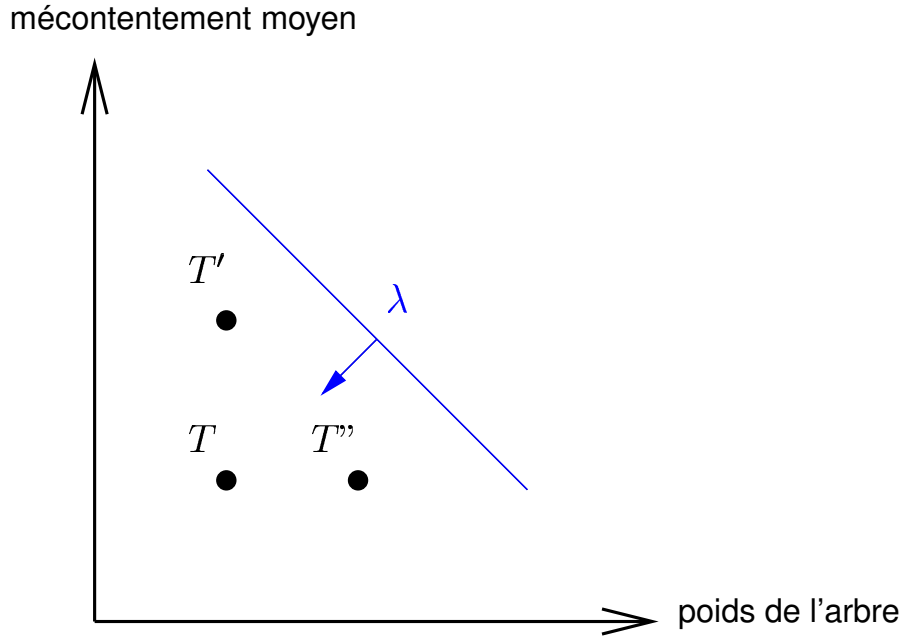


Minimiser le mécontentement moye

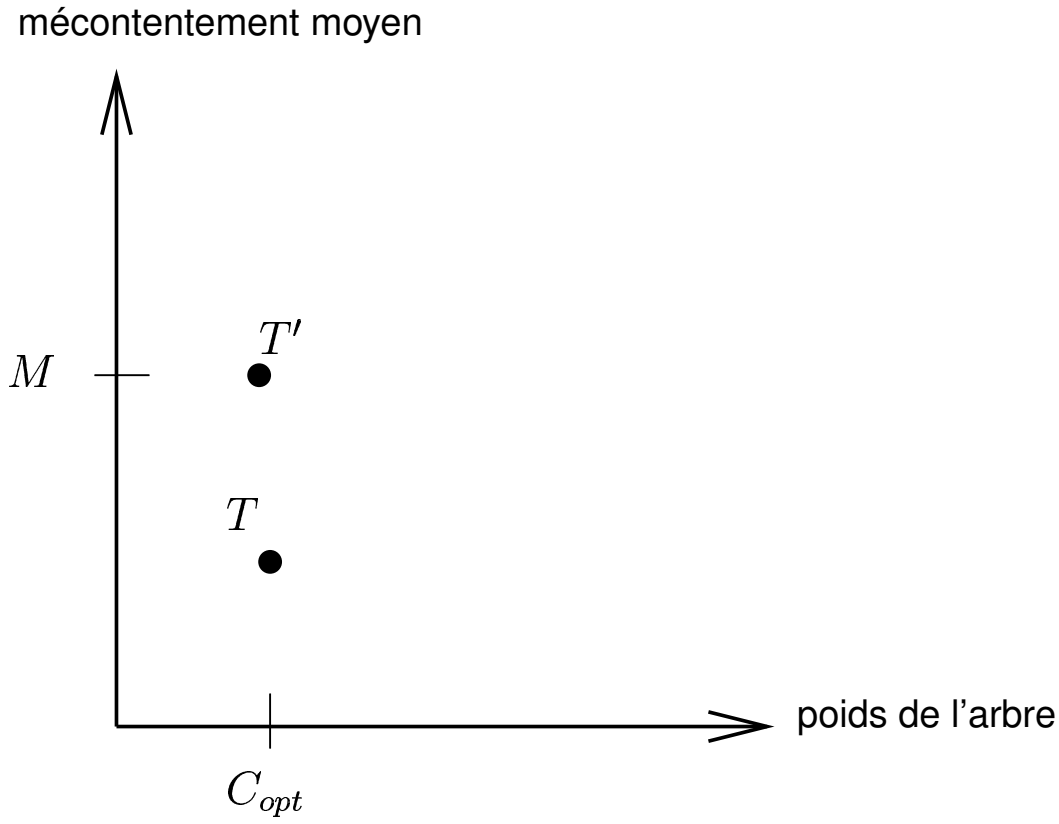
Utilisation d'un coût composite $\tilde{c}$, combinaison des deux critères :

$$\tilde{c}_{(u,v)} = \lambda c_{(u,v)} + (1 - \lambda)\frac{c_{(u,v)}}{F_{min}(v)}$$

Déterminer un λ tel que tout ACPM vis à vis de $\tilde{c}$ soit l'optimum recherché

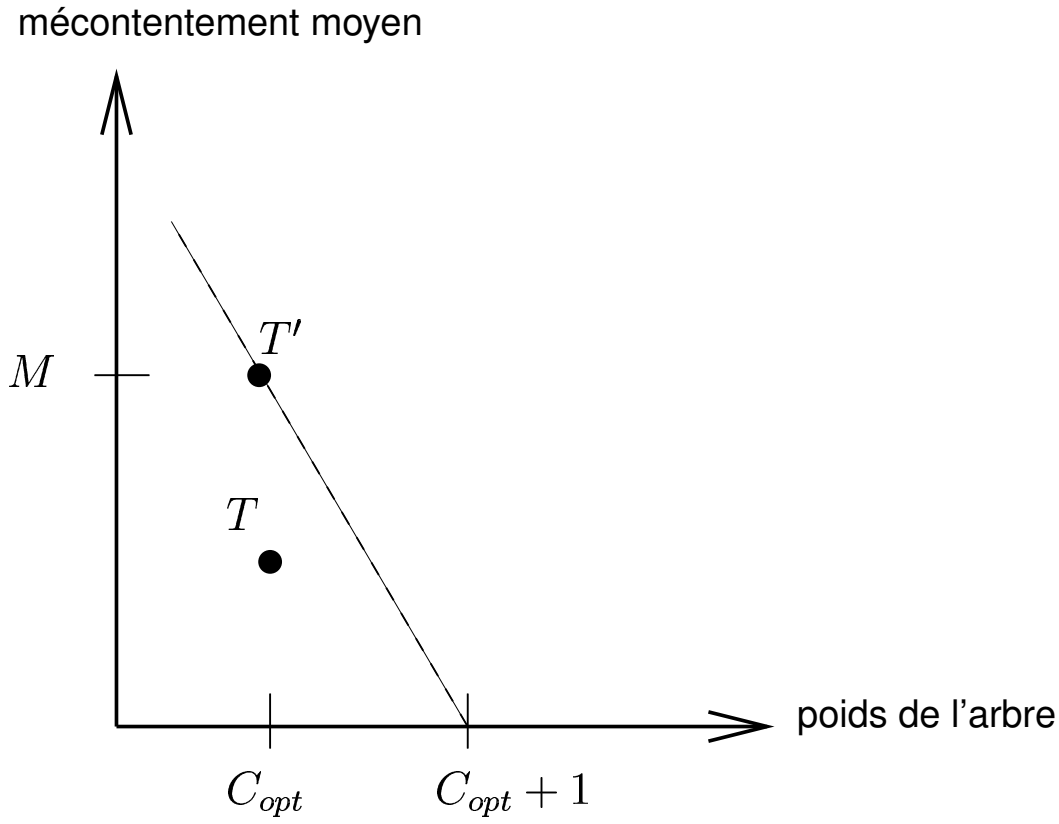


Minimiser le mécontentement moye



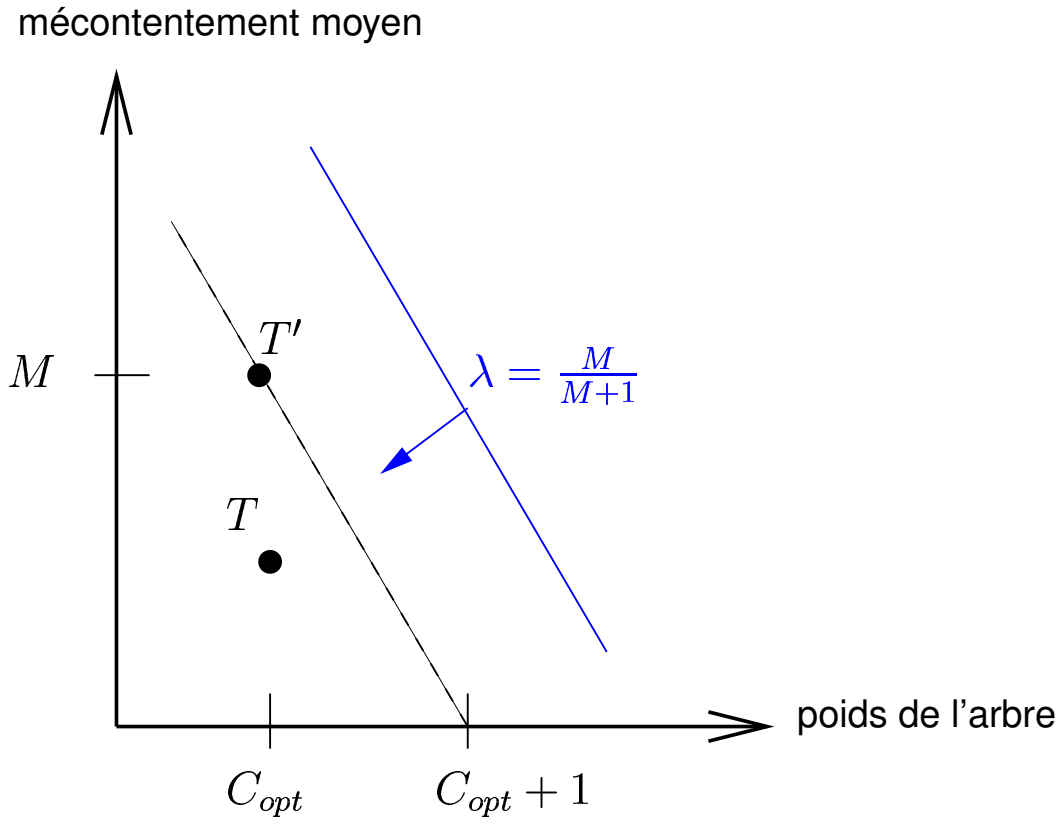
Déterminer un ACPM T'
Soit C_{opt} son poids et M son mécontentement moyen

Minimiser le mécontentement moye



L'optimum recherché est nécessairement sous la ligne discontinue

Minimiser le mécontentement moye



Poser $\lambda = \frac{M}{M+1}$

Conclusion

- Mécontentement : notion d'équité pour une allocation de coût
- Application au jeu de l'ACPM : 2 algorithmes
  - Minimiser le pire mécontentement
  - Minimiser le mécontentement moyen

Perspectives

- Algorithme distribué pour déterminer les optima
- Jeu de l'ACPM : ne plus se restreindre aux allocations de Bird
- Notion de mécontentement pour d'autres problèmes

À la recherche d'une solution "stable" : Deux problèmes mélant RO et Théorie des jeux

Eric Angel, Evripidis Bampis, Lélia Blin, Laurent Gourvès, Fanny Pascual

LaMI, Université d'Évry

Ordonnancement de tâches sur deux machines et stabilité approchée

Introduction

n tâches (agents) à ordonnancer sur 2 machines.

- *Fonction objectif individuel :*

  But de chaque agent : être exécuté le plus rapidement possible.

- *Fonction objectif global :*

  Notre but : minimiser le makespan (date de fin de la dernière tâche).

Introduction

- Chaque tâche connaît la longueur et la stratégie des autres tâches, ainsi que le protocole qui exécute les tâches.
- Chaque tâche choisit sur quelle machine elle va s'exécuter.
- pas de coopération entre tâches.

But : Obtenir un ordonnancement stable qui minimise le makespan.

Plan

- Introduction
- Protocole
  - Sans protocole : prix de l'anarchie
  - Mécanismes de coordination
  - Avec Protocole : prix de la stabilité
  - Prix de la stabilité α-approchée

Plan

- Introduction
- Protocole
  - Sans protocole : prix de l'anarchie
  - Mécanismes de coordination
  - Avec Protocole : prix de la stabilité
  - Prix de la stabilité α-approchée
- Prix de la stabilité α-approchée
  - Borne d'inapproximabilité
  - Borne inférieure
  - Borne supérieure
- Conclusion et perspectives

Sans protocole :

Définition : [Koutsoupias et Papadimitriou, STACS 1999]

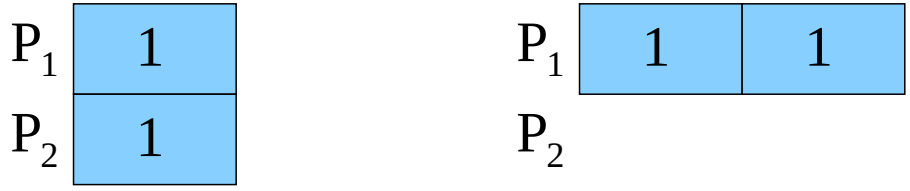
Prix de l'anarchie = $\dfrac{\text{Valeur du makespan ds le pire eq. de Nash}}{\text{Valeur du makespan dans OPT}}$.

Théorème :

Prix de l'anarchie $\geq \frac{3}{2}$.

Démonstration : 2 tâches : 

stratégie de châque tâche= proba de $\frac{1}{2}$ pour aller sur P1.
proba de $\frac{1}{2}$ pour aller sur P2.



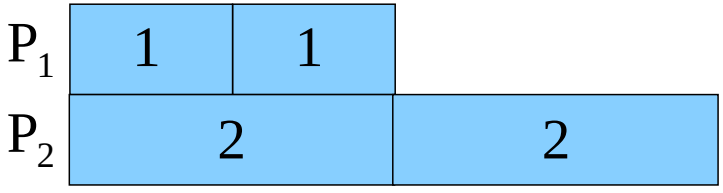
Espérance du makespan = $\frac{3}{2}$.

Mécanismes de coordination

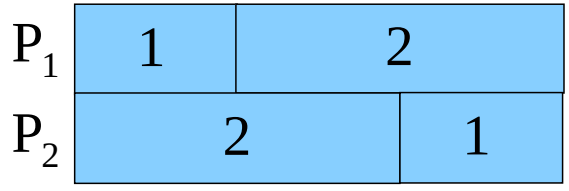
Définition :
Une politique par machine (avec délais eventuels entre les tâches).

Exemple : [Christodoulou et al, ICALP 2004]
Politique de la 1ère machine = SPT (de la plus petite tâche à la plus grande).

Politique de la 2ème machine =LPT (de la plus grande tâche à la plus petite).



Eq. de Nash



OPT

Mécanismes de coordination

Exemple :
Politique de chaque machine = LPT + petits délais entre tâches.
$\rightarrow$ Rapport d'approximation= 7/6.

Conjecture : [Christodoulou et al, ICALP 2004]
Il n'existe pas de meca. de coord. avec un rapport d'approximation inférieur à 7/6 pour notre problème.

Avec protocole :

On a :

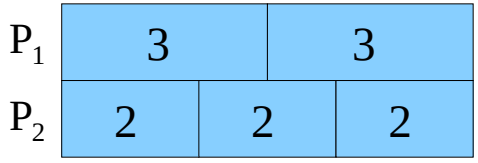
- Une politique par machine.
- Un protocole qui propose une affectation des tâches.

Les tâches acceptent ou refusent cette affectation.

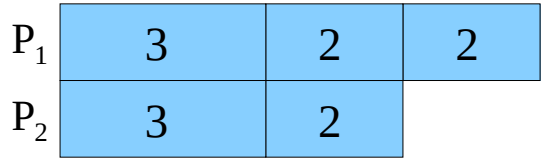
But : Avoir un protocole qui donne une solution :

- qui minimise le makespan
- et qui soit stable.

Exemple : Si chaque machine a comme politique LPT :



Pas stable



Stable

Prix de la stabilité

Introduit dans [Anshelevich et al, FOCS 2004].

Définition :

$$\text{Prix de la stabilité} = \frac{\text{Valeur du makespan ds le meilleur eq. de Nash}}{\text{Valeur du makespan dans OPT}}.$$

Rappel :

$$\text{Prix de l'anarchie} = \frac{\text{Valeur du makespan ds le pire eq. de Nash}}{\text{Valeur du makespan dans OPT}}.$$

Exemple : Si chaque machine a comme politique LPT, alors le prix de la stabilité est 7/6.

Prix de la stabilité α-approchée

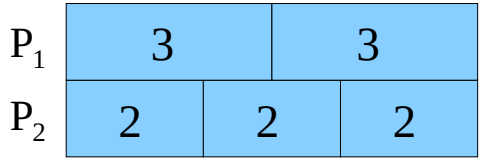
Equilibre de Nash α-approché= solution ds laquelle chaque agent n'augmente pas son profit plus de α fois en changeant de stratégie.

Définition :

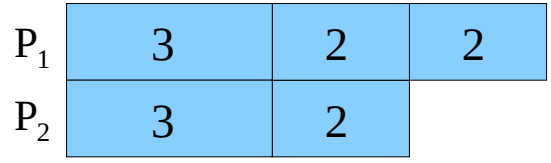
Prix de la stabilité α-approchée=

$$\frac{\text{Valeur du makespan ds le meilleur eq. de Nash } \alpha\text{-approché}}{\text{Valeur du makespan dans OPT}}.$$

Exemple : Chaque machine a comme politique LPT.

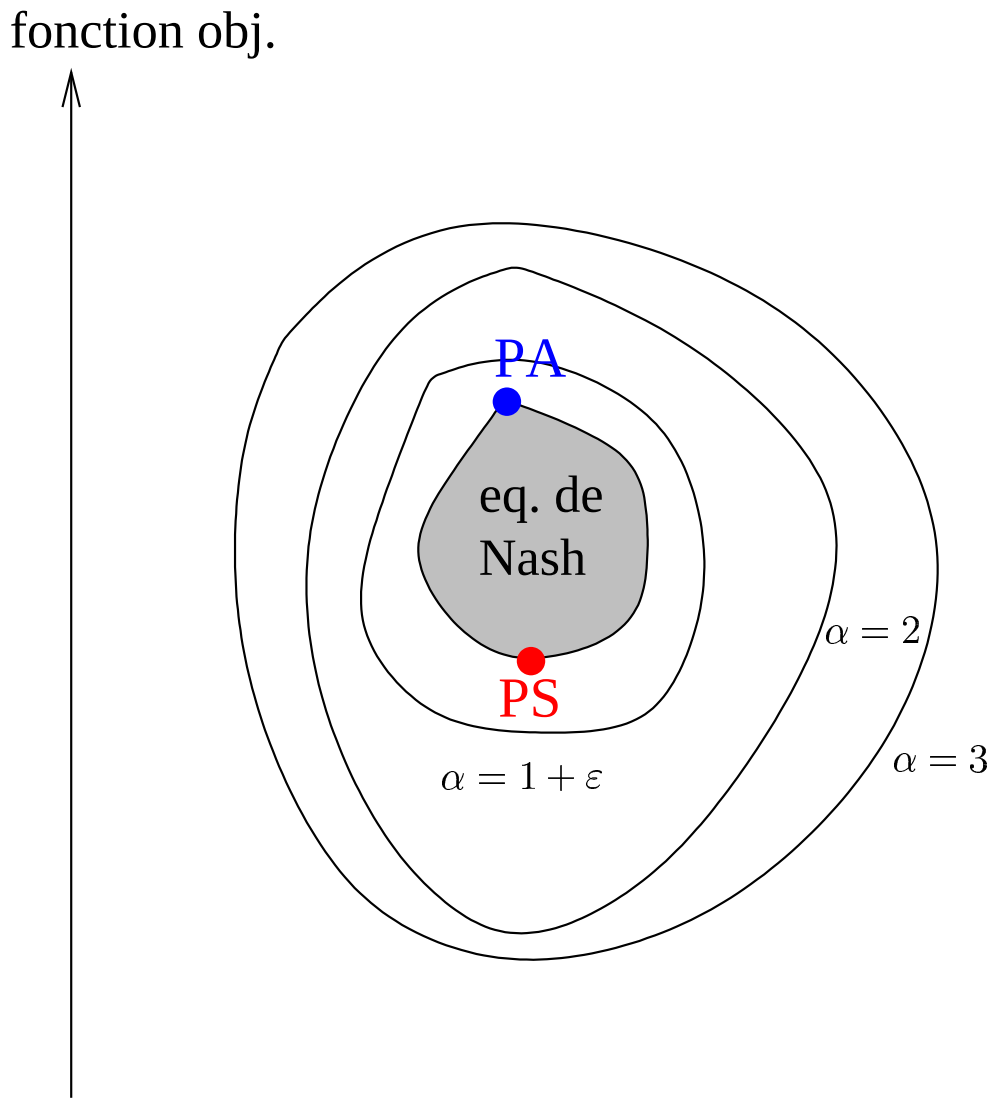


Eq. Nash 2-approché



Eq. de Nash

Prix de la stabilité α-approchée



Plan

- Introduction
- Protocole
  - Sans protocole : prix de l'anarchie $\longrightarrow$ Rapport = 3/2
  - Mécanismes de coordination $\longrightarrow$ Rapport $\geq$ 7/6
  - Avec Protocole : prix de la stabilité $\longrightarrow$ Rapport = 7/6 (LI
  - Prix de la stabilité α-approchée $\longrightarrow$ Rapport = $f(\alpha)$?
- Prix de la stabilité α-approchée
  - Borne d'inapproximabilité
  - Borne inférieure
  - Borne supérieure
- Conclusion et perspectives

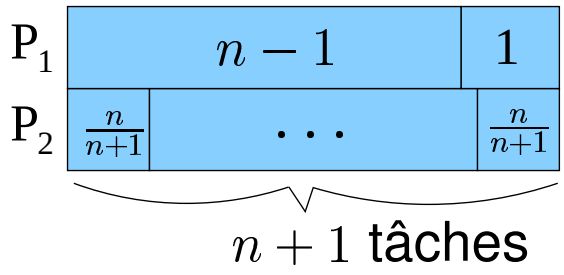
Borne d'inapproximabilité

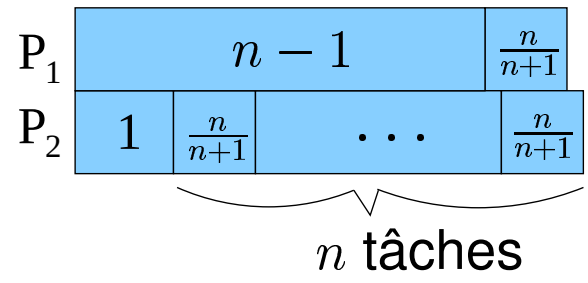
Théorème :

On a : n tâches. Soit $\varepsilon > 0$ t.q $\varepsilon = \frac{1}{n(n+1)}$. Politique des machines = LPT.

Alors : Il n'existe pas d'algo. avec un rapport d'approximation < $(1+\varepsilon)$ qui retourne des équilibres de Nash α-approchés, avec $\alpha < n$.

Démonstration :





Eq. Nash n-approché

Rapport d'approx = $1+\varepsilon$

Borne inférieure

Théorème :

Politique des machines = LPT.

Le prix de la stabilité α-approchée est $\leq \frac{8}{7}$, $\forall\ \alpha \geq 3$

Démonstration :

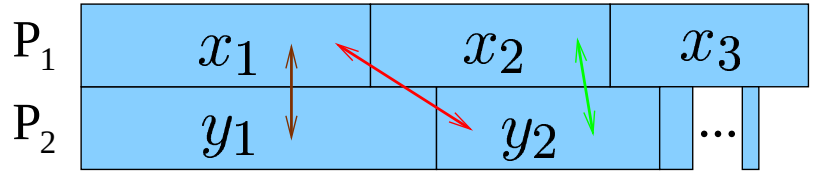
Algorithme LPT_{swap} :

$\frac{8}{7}$-approché et retourne des équilibres de Nash

3-approchés.

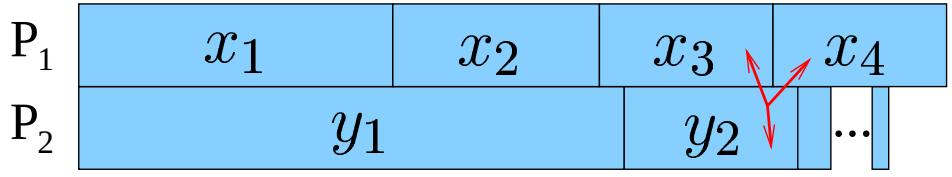
Borne inférieure : LPT_{swap}

- Construire un ordo. LPT
- Etudier cet ordo :
  - 1er cas :



Retourner le meilleur ordo parmi les 4 possibles.

  - 2ème cas :



Retourner le meilleur ordo parmi les 2 possibles.

  - Autres cas :
    Retourner LPT.

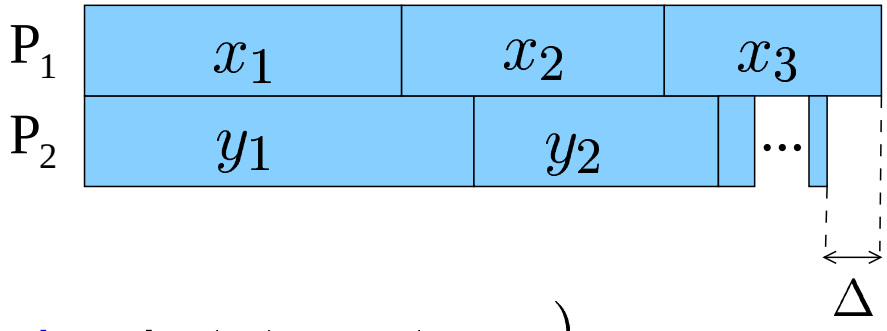
Borne inférieure : LPT_{swap}

Théorème 1 :
LPT_{swap} est $\frac{8}{7}$-approché.

Preuve :
Idée de la preuve :

Si l'ordo retourné par LPT n'est pas $\frac{8}{7}$-approché :
- On a 3 ou 4 tâches sur la machine la plus chargée.
- LPT_{swap} transforme cet ordo. en un ordo. $\frac{8}{7}$-approché.

Définition :



$x_3 = t_m{}^{ax}$

t_i est grande si $l(t_i) \geq l(t_m{}^{ax})$.

Borne inférieure : LPT_{swap}

Preuve : Soit un ordo LPT qui n'est pas $\frac{8}{7}$-approché.

Il y a exactement 2 grandes tâches sur P2 :

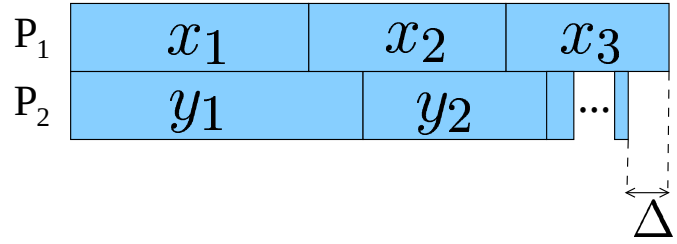
$$\left.\begin{array}{r} C_{max} = \frac{\sum_1^n l(t_i)+\Delta}{2} > \frac{8}{7}OPT \\ OPT \geq \frac{\sum_1^n l(t_i)}{2} \end{array}\right\} \Rightarrow \Delta > \frac{2}{7}OPT.$$

$\Rightarrow l(t_{max}) \geq \Delta$

$\Rightarrow$ la dernière tâche de P2 finit $\leq OPT - (\frac{1}{7} + \varepsilon)OPT$.

$\Rightarrow$ le nb max. de grandes tâches sur P2 est :

$\lfloor \frac{OPT-(\frac{1}{7}+\varepsilon)OPT}{\frac{2}{7}+\varepsilon'} \rfloor = 2.$



Borne inférieure : LPT_{swap}

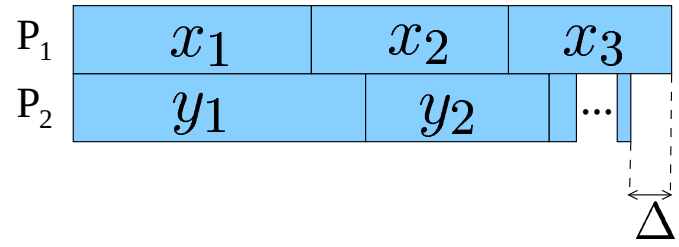
Il y a au plus 4 tâches sur P1 :
même arguments.

Il y a au moins 3 tâches sur P1 :
sinon on aurait un ordo. optimal.

Borne inférieure : LPT_{swap}

Il y a au plus 4 tâches sur P1 :
même arguments.

Il y a au moins 3 tâches sur P1 :
sinon on aurait un ordo. optimal.



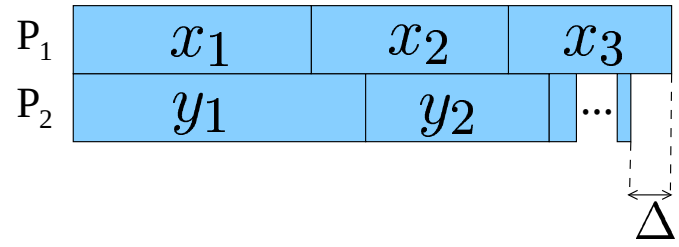
LPT_{swap} est un algo $\frac{8}{7}$-approché.

Sur P2 : il y a 2 grandes tâches et un ensemble de petites tâches.
$l(t_m{}^{ax}) \leq \frac{\frac{7}{6}OPT}{3} = \frac{7}{18}OPT.$
Petites tâches : commencent après t_{max} et avant $C_{max} - \Delta$
$\Rightarrow \sum_{t_i petite} l(t_i) \leq \frac{7}{18}OPT - \Delta \leq (\frac{7}{18} - \frac{2}{7})OPT < \frac{1}{7}OPT.$

Borne inférieure : LPT_{swap}

Il y a au plus 4 tâches sur P1 :
même arguments.

Il y a au moins 3 tâches sur P1 :
sinon on aurait un ordo. optimal.



LPT_{swap} est un algo $\frac{8}{7}$-approché.

Sur P2 : il y a 2 grandes tâches et un ensemble de petites tâches.
$l(t_m{}^{ax}) \leq \frac{\frac{7}{6}OPT}{3} = \frac{7}{18}OPT.$
Petites tâches : commencent après t_{max} et avant $C_{max} - \Delta$
$\Rightarrow \sum_{t_i petite} l(t_i) \leq \frac{7}{18}OPT - \Delta \leq (\frac{7}{18} - \frac{2}{7})OPT < \frac{1}{7}OPT.$

Soit OPT' un ordo optimal des grandes tâches : $OPT' \leq OPT$ et on obtient OPT' avec LPT_{swap}.

Borne inférieure : LPT_{swap}

Théorème 1 :
LPT_{swap} est $\frac{8}{7}$-approché.

Théorème 2 :
LPT_{swap} retourne des équilibres de Nash 3-approchés.

Idée de la preuve : cas par cas.

Corollaire :
Le prix de la stabilité α-approchée est $\leq \frac{8}{7},\ \forall \alpha \geq 3$.

Plan

- Introduction
- Protocole
  - Présentation du problème
  - Sans protocole : prix de l'anarchie
  - Mécanismes de coordination
  - Avec Protocole : prix de la stabilité
  - Prix de la stabilité α-approchée
- Prix de la stabilité α-approchée
  - Borne d'inapproximabilité
  - Borne inférieure
  - Borne supérieure
- Conclusion et perspectives

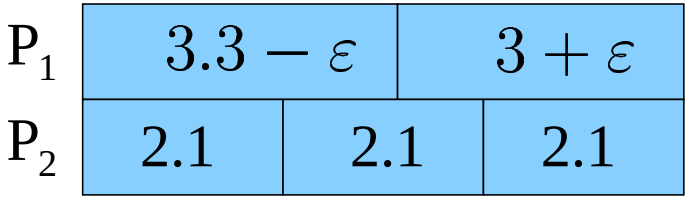
Borne supérieure

Théorème :

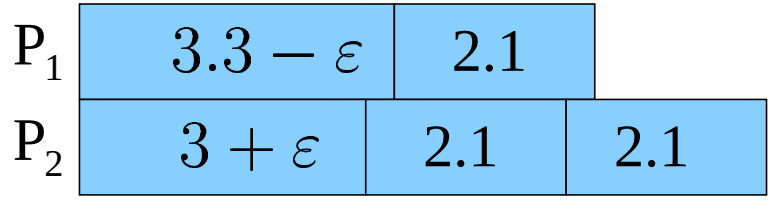
Soit $\varepsilon > 0$. Politique des machines = LPT.

Le prix de la stabilité α-approchée est $\geq \frac{8}{7}$, $\forall\ \alpha < 2.1$

Démonstration :



Makespan = 6.3

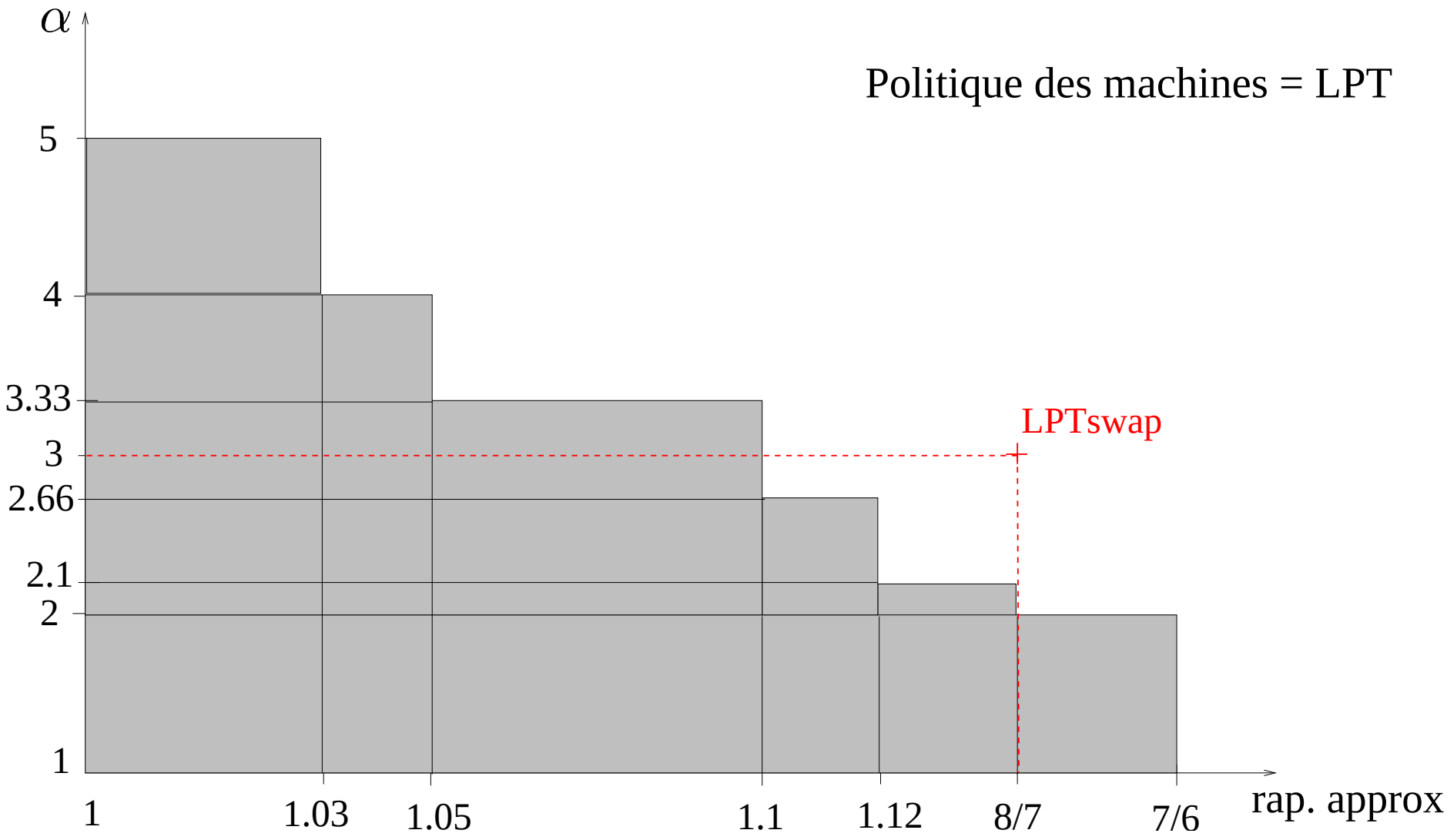


Makespan = 7.2

Eq. Nash α-approché avec $\alpha = \frac{6.3}{3+\varepsilon} > 2.1 - \varepsilon$

Rapport d'approx. $= \frac{7.2+\varepsilon}{6.3} > \frac{8}{7}$

Bilan



Politique des machines = LPT

Conclusion et perspectives

Conclusion :

- $\alpha \to \infty$ qd PS α-app.$\to 1$.
- PS α-app. $\leq 8/7$, $\forall\ \alpha \geq 3$
  $\geq 8/7$, $\forall\ \alpha < 2.1$.
- Algorithme LPT_{swap}.

Conclusion et perspectives

Conclusion :

- $\alpha \to \infty$ qd PS α-app.$\to 1$.
- PS α-app. $\leq 8/7$, $\forall\ \alpha \geq 3$
  $\geq 8/7$, $\forall\ \alpha < 2.1$.
- Algorithme LPT_{swap}.

Perspectives :

- Analyse plus fine du PS α-app. ; autres algorithmes ?
- Une autre politique meilleure que LPT ?
- Etudier le prix de la stabilité approchée dans d'autres problèmes.