ECOLE DES MINES DE NANTES

irin

Des explications “opérationnelles” pour les contraintes globales

GT Contraintes et RO

Guillaume Rochart

guillaume.rochart@emn.fr

Dépt Informatique, École des Mines de Nantes

IRIN, Université de Nantes

Introduction

- Recherche opérationnelle et programmation par contraintes traitent des sujets proches
- PPC : apporte une modélisation “haut-niveau”
  - langage,
  - modulaire/réutilisable,
  - compositions de solveurs. . .
- RO : apporte des algorithmes efficaces

Introduction

- Recherche opérationnelle et programmation par contraintes traitent des sujets proches
- PPC : apporte une modélisation “haut-niveau”
  - langage,
  - modulaire/réutilisable,
  - compositions de solveurs. . .
- RO : apporte des algorithmes efficaces

- Comment tirer partie de l’efficacité de la R.O. dans le modèle CSP ?

ECOLE DES MINES DE NANTES

Introduction

- Recherche opérationnelle et programmation par contraintes traitent des sujets proches
- PPC : apporte une modélisation “haut-niveau”
  - langage,
  - modulaire/réutilisable,
  - compositions de solveurs. . .
- RO : apporte des algorithmes efficaces

- Comment tirer partie de l’efficacité de la R.O. dans le modèle CSP ?
  - La R.O. pour *résoudre* les contraintes
  - La R.O. pour *modéliser* les contraintes

  $\Rightarrow$ La R.O. pour *expliquer* les contraintes

Le formalisme CSP

On peut définir un Problème de Satisfaction de Contraintes par le triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ avec :

- $\mathcal{X}$ un ensemble de variables,
- $\mathcal{D}$ un ensemble de domaines tels que $\mathcal{D}_i$ soit le domaine de $\mathcal{X}_i$,
- $\mathcal{C}$ un ensemble de “*contraintes*” sur ces variables.

Le formalisme CSP

On peut définir un Problème de Satisfaction de Contraintes par le triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ avec :

- $\mathcal{X}$ un ensemble de variables,
- $\mathcal{D}$ un ensemble de domaines tels que $\mathcal{D}_i$ soit le domaine de $\mathcal{X}_i$,
- $\mathcal{C}$ un ensemble de “*contraintes*” sur ces variables.

Le problème : déterminer une affectation respectant $\mathcal{D}$ des variables $\mathcal{X}$ telle que les contraintes $\mathcal{C}$ soient respectées.

Le formalisme CSP

On peut définir un Problème de Satisfaction de Contraintes par le triplet $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ avec :

- $\mathcal{X}$ un ensemble de variables,
- $\mathcal{D}$ un ensemble de domaines tels que $\mathcal{D}_i$ soit le domaine de $\mathcal{X}_i$,
- $\mathcal{C}$ un ensemble de “*contraintes*” sur ces variables.

Le problème : déterminer une affectation respectant $\mathcal{D}$ des variables $\mathcal{X}$ telle que les contraintes $\mathcal{C}$ soient respectées.

Dans le cas général, il s’agit d’un problème NP difficile.

La “R.O.” pour résoudre (1/4)

Les contraintes peuvent être implémentées comme :

- des *relations*, c’est-à-dire des tuples respectant la contrainte
- des *fonctions de réductions de domaine*,
- des procédures événementielles.

La “R.O.” pour résoudre (1/4)

Les contraintes peuvent être implémentées comme :

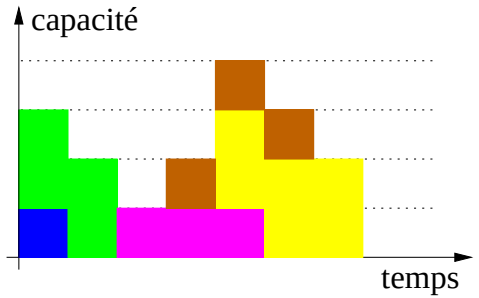
- des *relations*, c’est-à-dire des tuples respectant la contrainte
- des *fonctions de réductions de domaine*,
- des procédures événementielles.

Les contraintes ont donc besoin d’algorithmes :

- pour déterminer si une contrainte est vérifiée ou non,
- pour déduire des réductions de domaines en fonction des domaines ou des événements qu’elles reçoivent.

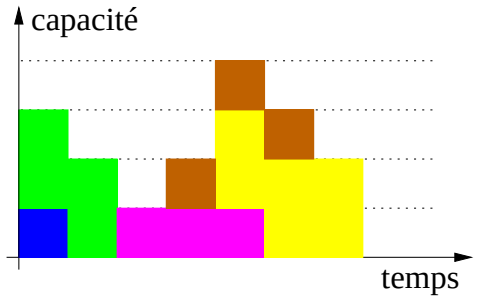
La “R.O.” pour résoudre (2/4)

- `cumulative` :

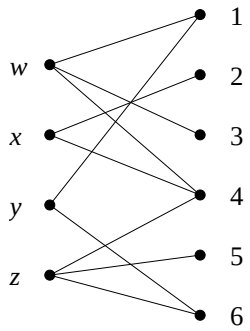


La “R.O.” pour résoudre (2/4)

- `cumulative` :

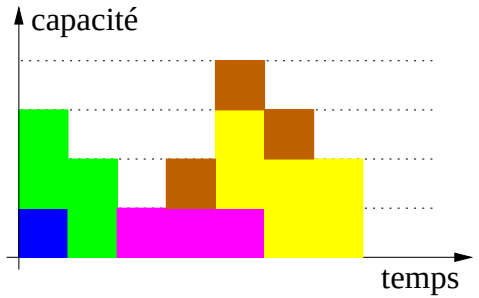


- `alldiff` :

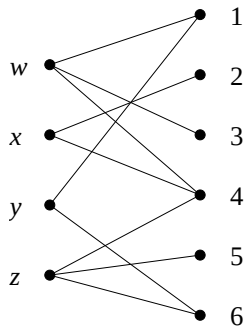


La “R.O.” pour résoudre (2/4)

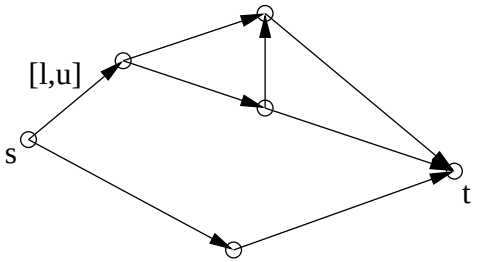
- `cumulative` :



- `alldiff` :



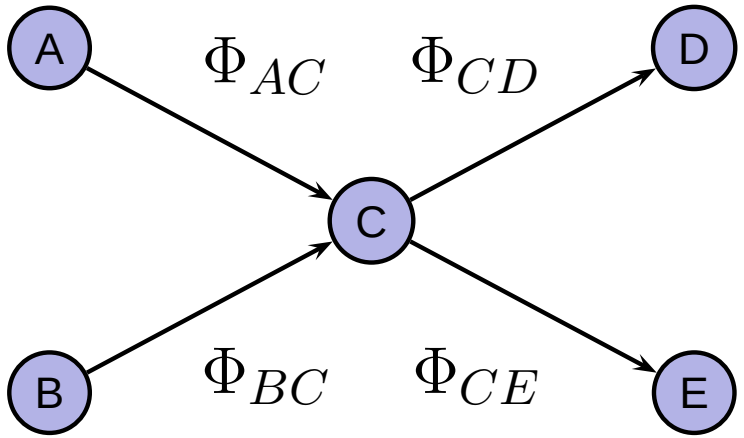
- `flow` :



La “R.O.” pour résoudre (3/4)

La contrainte de flot utilisée par [Benoist et al., 2002] permet de garantir deux propriétés :

- La conservation du flot aux noeuds (combinaison linéaire)



$$min(\Phi_{CD}) = min(\Phi_{AC}) + min(\Phi_{BC}) - max(\Phi_{CE})$$

La “R.O.” pour résoudre (3/4)

La contrainte de flot utilisée par [Benoist et al., 2002] permet de garantir deux propriétés :

- La conservation du flot aux noeuds (combinaison linéaire)
- Respect des domaines de variables :
  - pour chaque arc du réseau,
  - pour l’arc de retour représentant le flot global.

La “R.O.” pour résoudre (3/4)

La contrainte de flot utilisée par [Benoist et al., 2002] permet de garantir deux propriétés :

- La conservation du flot aux noeuds (combinaison linéaire)
- Respect des domaines de variables :
  - pour chaque arc du réseau,
  - pour l’arc de retour représentant le flot global.

$\Rightarrow$ Cette contrainte permet de modéliser des problèmes de partages d’activités ou de ressources.

La “R.O.” pour résoudre (4/4)

Deux phases principales :

- Initialisation de la contrainte :
  - Test de `faisabilité du flot` : transformation de [Berge, 1983]
  - Calcul du `flot maximal` : algorithme de préflot
  - Calcul du `flot minimal`

La “R.O.” pour résoudre (4/4)

Deux phases principales :

- Initialisation de la contrainte :
  - Test de `faisabilité du flot` : transformation de [Berge, 1983]
  - Calcul du `flot maximal` : algorithme de préflot
  - Calcul du `flot minimal`
- Propagation de la contrainte :
  - Propagation sur la conservation du flot
  - Propagation sur la `faisabilité du flot` : recherche d’un chemin
  - Maintien du `flot maximal`

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

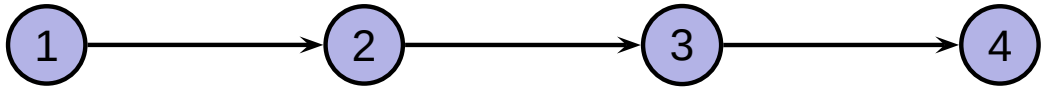
- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*

LOOP

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*

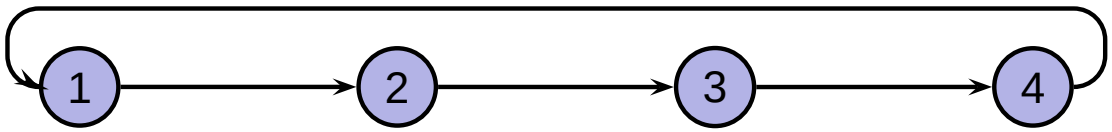


PATH

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*

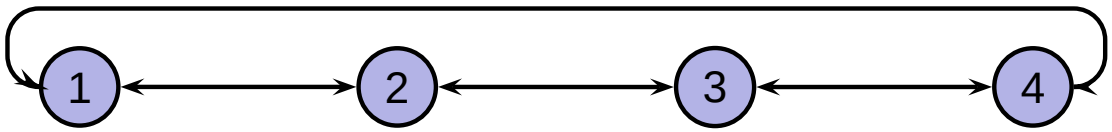


CIRCUIT

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*

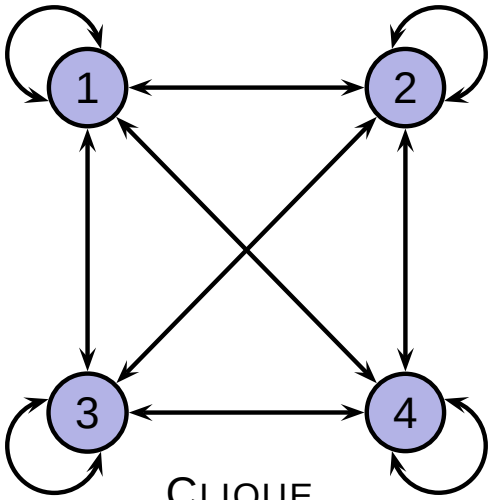


CYCLE

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*

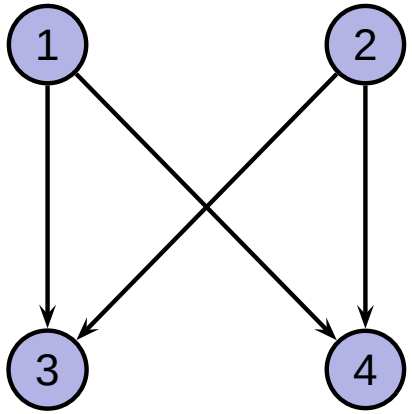


CLIQUE

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*



PRODUIT

La “R.O.” pour modéliser (1/2)

Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*
- Des *contraintes basiques* pour les arcs

La “R.O.” pour modéliser (1/2)

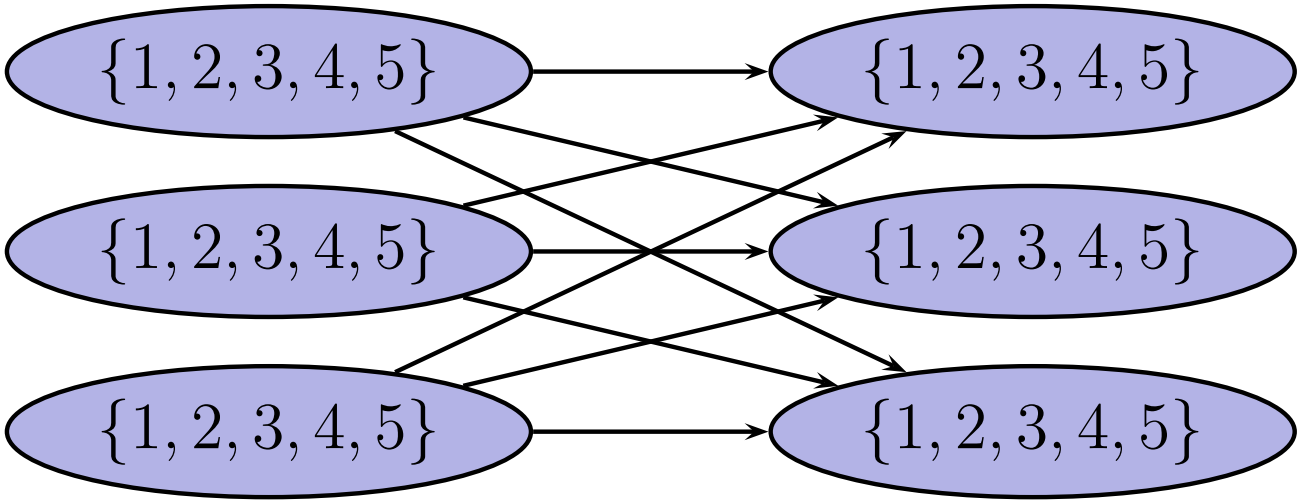
Pour modéliser une contrainte, [Beldiceanu, 2000] propose d’utiliser les caractéristiques suivantes :

- Un *générateur de sommets* : quasiment exclusivement IDENTITÉ, mais propose aussi PAIRE
- Un *générateur d’arcs*
- Des *contraintes basiques* pour les arcs
- Des contraintes sur des *propriétés* du graphe final : NVERTEX, NARC, NSOURCE, NCC, MAX_NSCC...

La “R.O.” pour modéliser (2/2)

La contrainte common contraint le nombre valeurs en commun entre deux ensemble de variables.

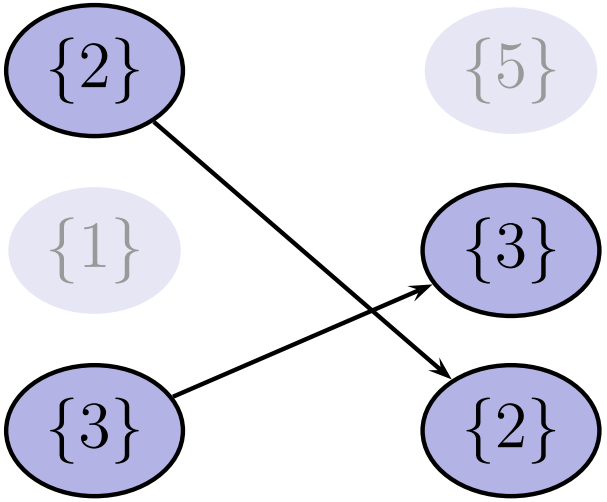
- Générateur de noeuds : IDENTITÉ
- Générateur d’arcs : PRODUIT
- Contrainte basique : $=$
- Propriété : NSOURCE $= \alpha \wedge$ NSINK $= \beta$



La “R.O.” pour modéliser (2/2)

La contrainte common contraint le nombre valeurs en commun entre deux ensemble de variables.

- Générateur de noeuds : IDENTITÉ
- Générateur d’arcs : PRODUIT
- Contrainte basique : $=$
- Propriété : NSOURCE $= \alpha \wedge$ NSINK $= \beta$



Explications

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Exemple Supposons trois variables et deux contraintes : $c_1 \equiv V_1 < V_2$ et $c_2 \equiv V_2 < V_3$.

	V_1	V_2	V_3
Borne Inf.	1	1	1
Borne Sup.	6	6	6

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Exemple Supposons trois variables et deux contraintes : $c_1 \equiv V_1 < V_2$ et $c_2 \equiv V_2 < V_3$.

	V_1	V_2	V_3
Borne Inf.	1	2	3
Borne Sup.	4	5	6

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Exemple Supposons trois variables et deux contraintes : $c_1 \equiv V_1 < V_2$ et $c_2 \equiv V_2 < V_3$.

	V_1	V_2	V_3
Borne Inf.	1	2	3
Borne Sup.	4	5	6

Prenons le choix $d_1 \equiv V_3 = 5$.

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Exemple Supposons trois variables et deux contraintes : $c_1 \equiv V_1 < V_2$ et $c_2 \equiv V_2 < V_3$.

	V_1	V_2	V_3
Borne Inf.	1	2	5
Borne Sup.	4	5	5

Prenons le choix $d_1 \equiv V_3 = 5$.

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Exemple Supposons trois variables et deux contraintes : $c_1 \equiv V_1 < V_2$ et $c_2 \equiv V_2 < V_3$.

	V_1	V_2	V_3
Borne Inf.	1	2	5
Borne Sup.	4	4	5

Prenons le choix $d_1 \equiv V_3 = 5$.

$$expl(V_2 \leq 4) = \{d_1, c_2\}$$

Les explications

Définition (Explication) L’explication d’un domaine ou d’une inférence (contradiction, réduction de domaine, instanciation...) noté $\mathcal{X}$ est un sous-ensemble $\mathcal{E}$ de contraintes (du système et de choix) tel que :

$$c_1 \wedge c_2 \ldots \wedge d_1 \wedge d_2 \ldots \Rightarrow \mathcal{X}$$

Exemple Supposons trois variables et deux contraintes : $c_1 \equiv V_1 < V_2$ et $c_2 \equiv V_2 < V_3$.

	V_1	V_2	V_3
Borne Inf.	1	2	5
Borne Sup.	3	4	5

Prenons le choix $d_1 \equiv V_3 = 5$.

$$expl(V_2 \leq 3) = \{d_1, c_1, c_2\}$$

ECOLE DES MINES DE NANTES

Application des explications

- Aide au développement
- De nombreux algorithmes utilisent cette notion d’explication :
  - `BackJumping`,
  - `Dynamic Backtracking`, et `MAC-DBT`,
  - `Decision Repair`...

Application des explications

- Aide au développement
- De nombreux algorithmes utilisent cette notion d’explication :
  - `BackJumping`,
  - `Dynamic Backtracking`, et `MAC-DBT`,
  - `Decision Repair`...
- L’application aux contraintes globales implique quelques difficultés :
  - Expliquer les décisions nécessitent l’accès aux données internes,
  - L’algorithme doit être décrémentable.

Application des explications

- Aide au développement
- De nombreux algorithmes utilisent cette notion d’explication :
  - `BackJumping`,
  - `Dynamic Backtracking`, et `MAC-DBT`,
  - `Decision Repair`...
- L’application aux contraintes globales implique quelques difficultés :
  - Expliquer les décisions nécessitent l’accès aux données internes,
  - L’algorithme doit être décrémentable.
- De bons résultats ont été obtenus avec `Stretch` ([Rochart and Jussien, 2003])

Explications “opérationnelles”

ECOLE DES MINES DE NANTES

Expliquer la contrainte de flot

- Les décisions à expliquer sont les décisions basées sur :
  - la maximisation du flot,
  - la minimisation du flot,
  - la faisabilité d’un flot (compatibilité).
- Il faut donc pouvoir expliquer chacune de ces propriétés.

Expliquer la contrainte de flot

- Les décisions à expliquer sont les décisions basées sur :
  - la maximisation du flot,
  - la minimisation du flot,
  - la faisabilité d’un flot (compatibilité).
- Il faut donc pouvoir expliquer chacune de ces propriétés.

$\Rightarrow$ La notion principale pour ces explications : la *coupe*

Les coupes(1/2)

Coupe

- On appelle *coupe de* G, un sous-ensemble S qui inclut la source s et pas le puits t ;

Les coupes(1/2)

Coupe

- On appelle *coupe de* G, un sous-ensemble S qui inclut la source s et pas le puits t ;
- On note $T = X - S$;
- Arcs *entrant* et arcs *sortant*.

Les coupes(1/2)

Coupe

- On appelle *coupe de* G, un sous-ensemble S qui inclut la source s et pas le puits t ;
- On note $T = X - S$;
- Arcs *entrant* et arcs *sortant*.

Capacité de coupe

- La capacité d’une coupe est définie par :

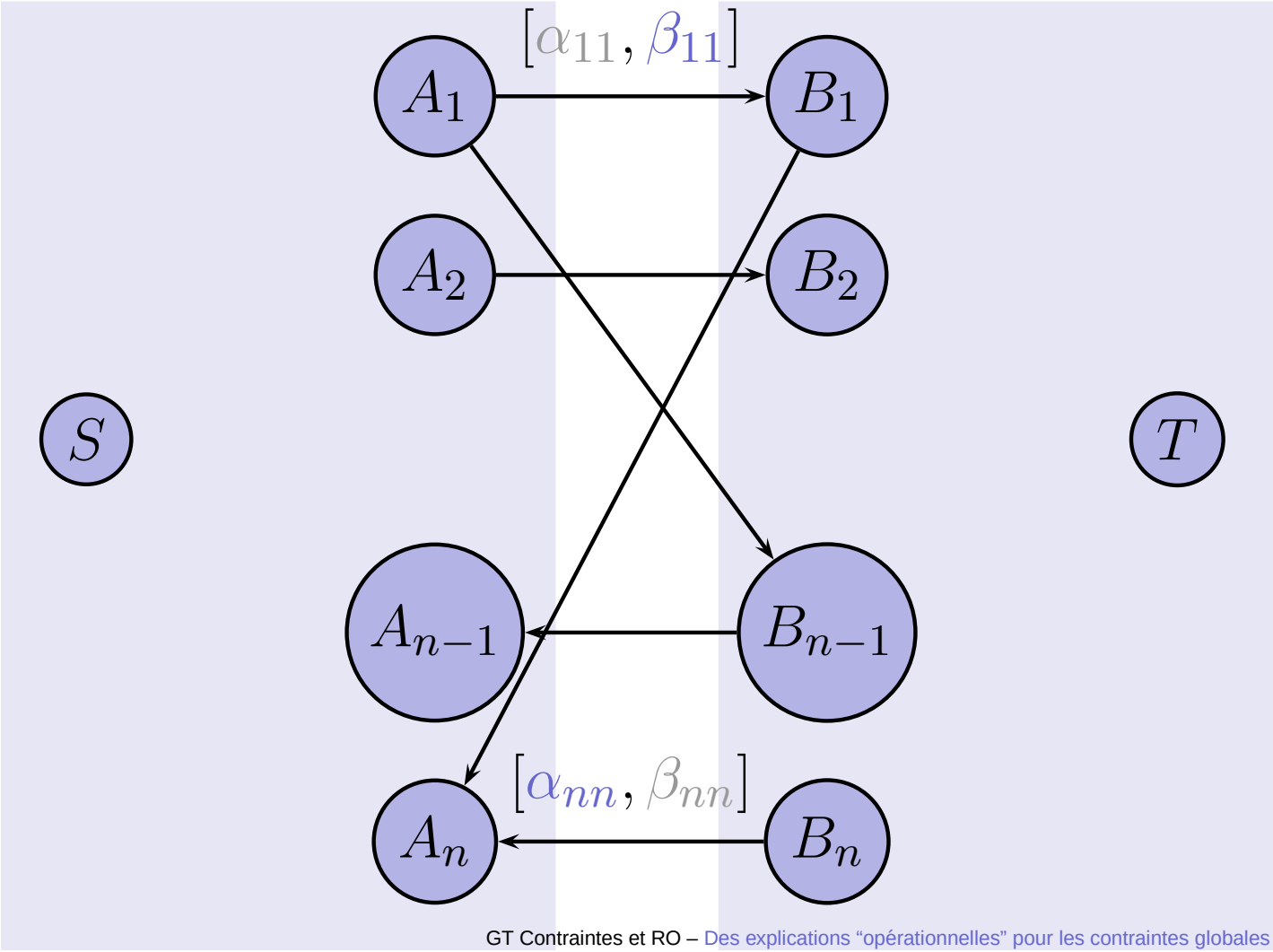
$$C(S,T) = \sum_{sortant} U_{ij} - \sum_{entrant} L_{ij}$$

Les coupes (2/2)

Propriété : Le débit de tout flot est inférieur ou égal à la capacité de toute coupe.

Les coupes (2/2)

Propriété : Le débit de tout flot est inférieur ou égal à la capacité de toute coupe.



Explication du flot maximal

- On souhaite trouver une explication montrant $F < \gamma$,
- Or, “le débit de tout flot est inférieur la capacité de toute coupe”,

Explication du flot maximal

- On souhaite trouver une explication montrant $F < \gamma$,
- Or, “le débit de tout flot est inférieur la capacité de toute coupe”,

- On a donc l’explication suivante :

$$expl(F \leq C(S,T)) = \bigcup_{sortant} expl(bsup(V_{ij})) \cup \bigcup_{entrant} expl(binf(V_{ij}))$$

Explication du flot maximal

- On souhaite trouver une explication montrant $F < \gamma$,
- Or, “le débit de tout flot est inférieur la capacité de toute coupe”,

- On a donc l’explication suivante :

$$expl(F \leq C(S,T)) = \bigcup_{sortant} expl(bsup(V_{ij})) \cup \bigcup_{entrant} expl(binf(V_{ij}))$$

- De plus, il s’agit de l’explication de la valeur maximale du débit si l’on prend une coupe minimale.

Explication du flot minimal

- Problème similaire à la maximisation
- On échange la source avec le puits

- On trouve la même explication :

$$expl(binf(F)) = \bigcup_{sortant} expl(bsup(V_{ij})) \cup \bigcup_{entrant} expl(binf(V_{ij}))$$

Explication du flot minimal

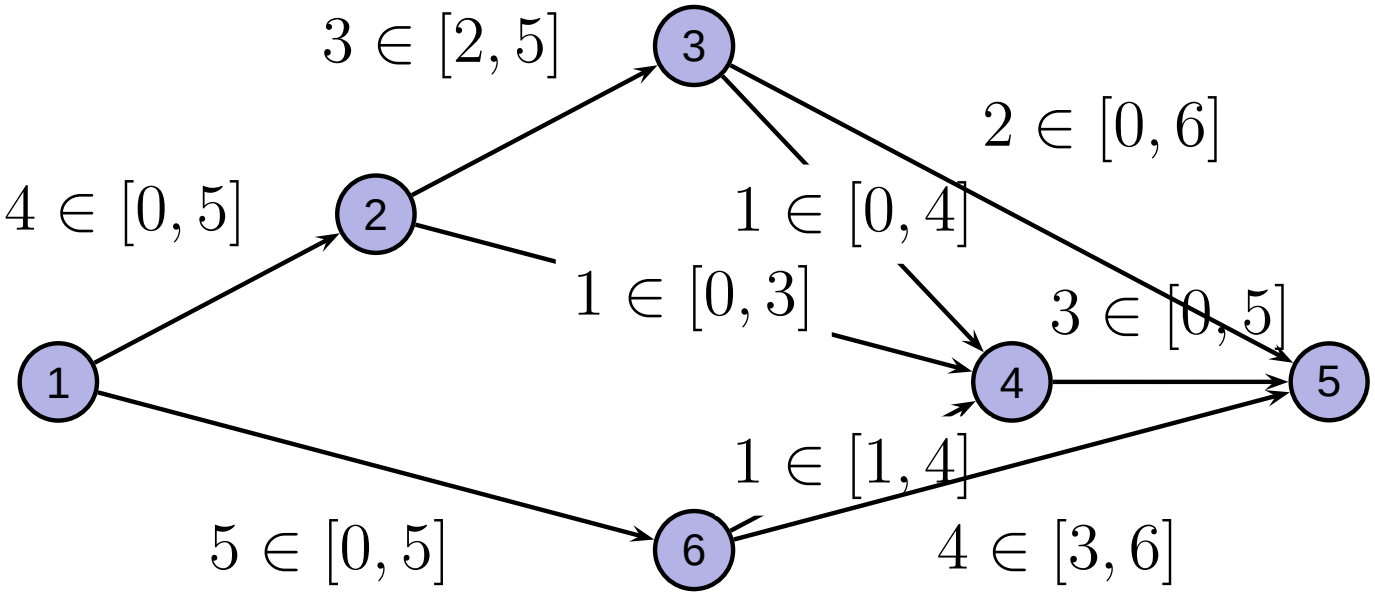
- Problème similaire à la maximisation
- On échange la source avec le puits

- On trouve la même explication :

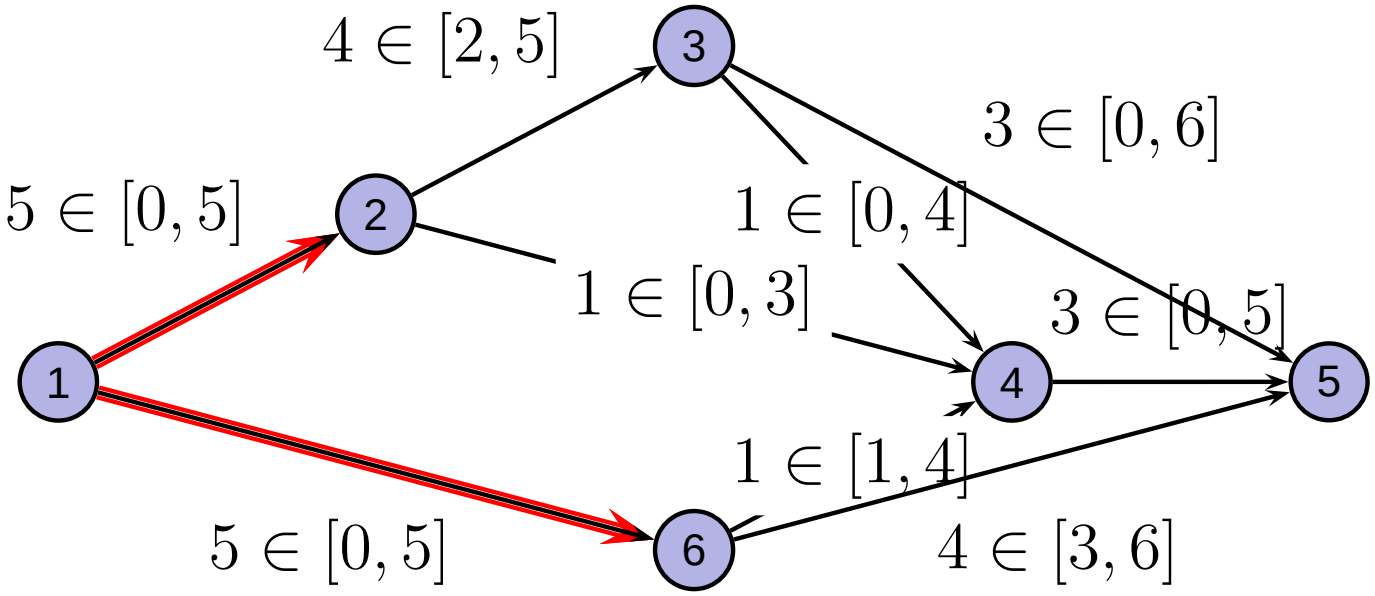
$$expl(binf(F)) = \bigcup_{sortant} expl(bsup(V_{ij})) \cup \bigcup_{entrant} expl(binf(V_{ij}))$$

- **Mais** la coupe est inversée (S contient le puits)

Exemple

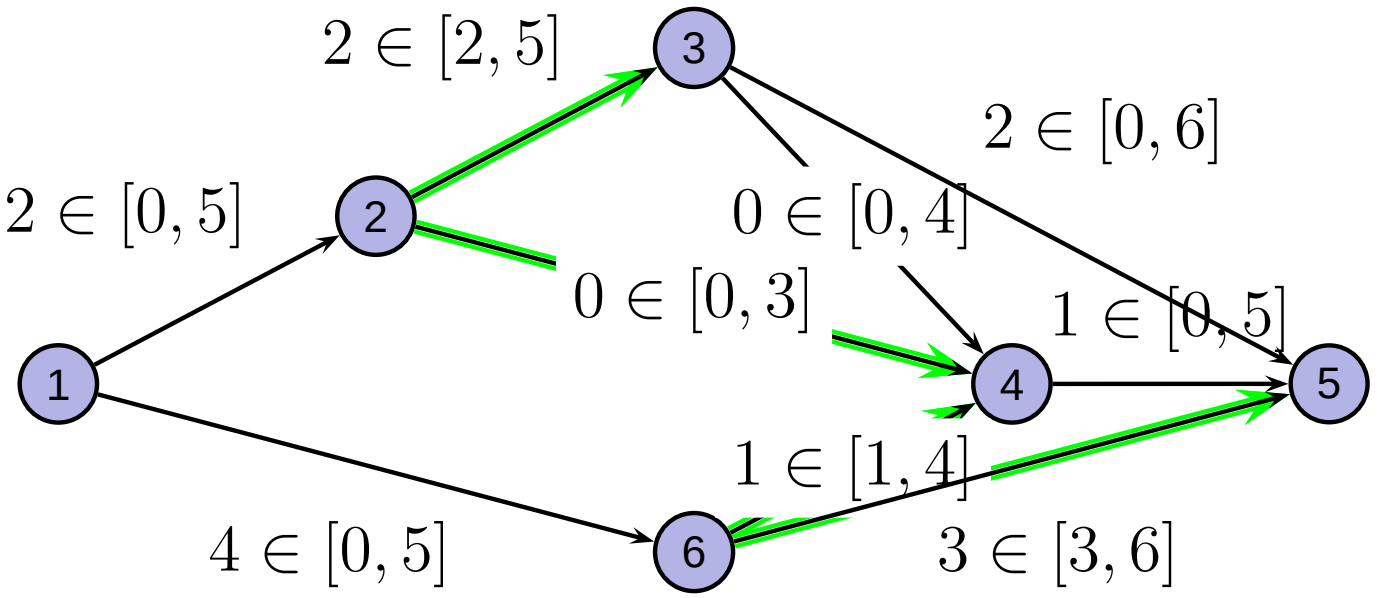


Exemple



$$expl(F \leq 10) = expl(bsup(V_{1,2})) \cup expl(bsup(V_{1,6}))$$

Exemple

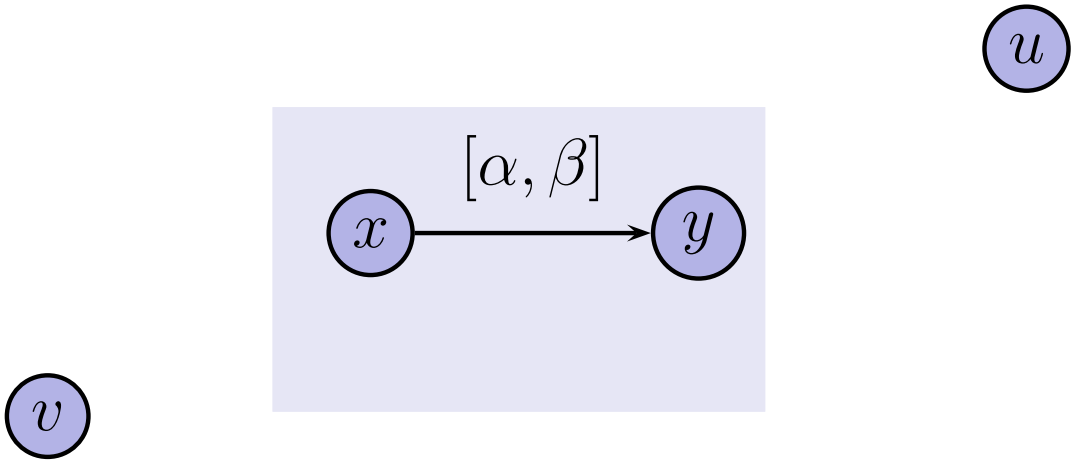


$$expl(F \leq 10) = expl(bsup(V_{1,2})) \cup expl(bsup(V_{1,6}))$$

$$\begin{aligned} expl(F \geq 6) \quad = \quad & expl(binf(V_{2,3})) \cup expl(binf(V_{2,4})) \\ & \cup expl(binf(V_{6,4})) \cup expl(binf(V_{6,5})) \end{aligned}$$

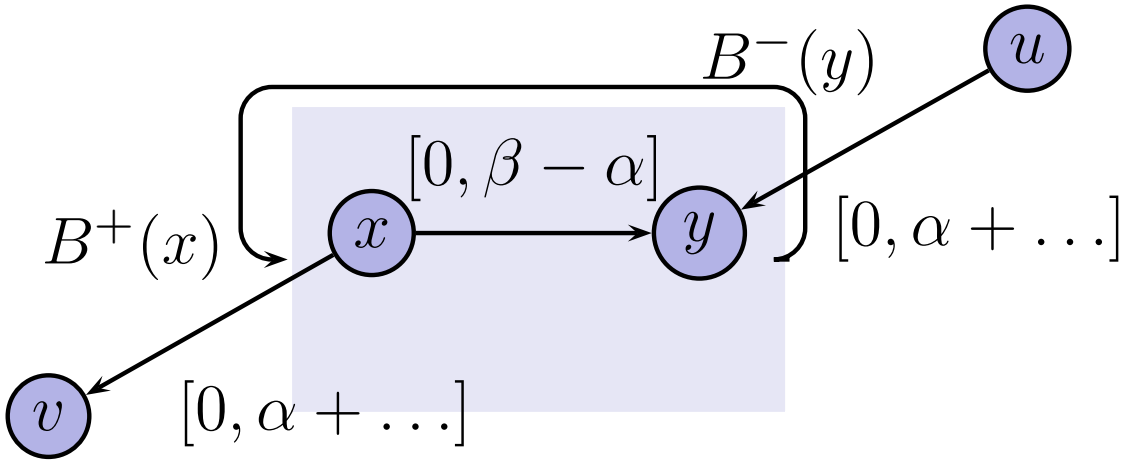
Explication du flot faisable

- Idée de [Berge, 1983] :



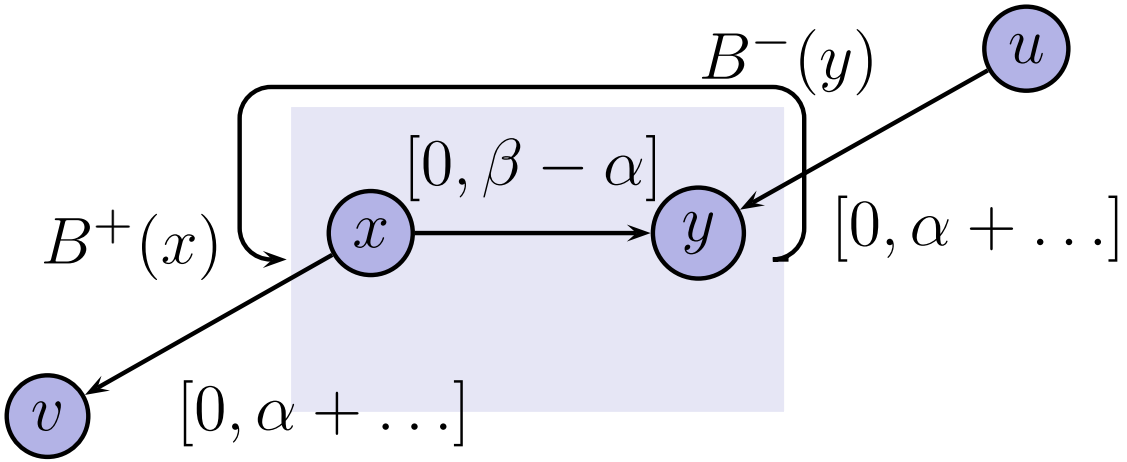
Explication du flot faisable

- Idée de [Berge, 1983] :



Explication du flot faisable

- Idée de [Berge, 1983] :

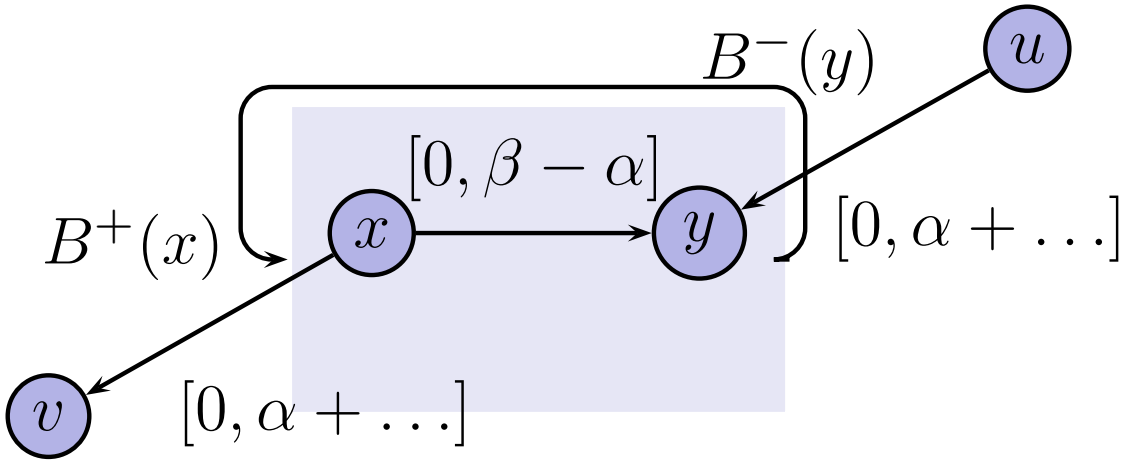


- En cas d’échec, on obtient une inégalité

$$F = C(S,T) < \sum_i B^+(i)$$

Explication du flot faisable

- Idée de [Berge, 1983] :

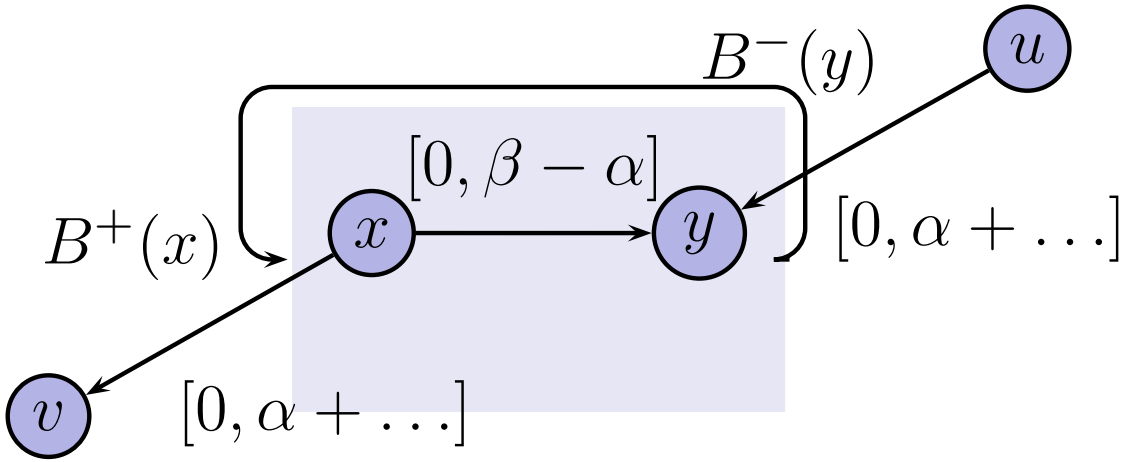


- En cas d’échec, on obtient une inégalité

$$\sum_C B^-(i) + \sum_C U'ij + \sum_C B^+(i) < \sum_i B^+(i)$$

Explication du flot faisable

- Idée de [Berge, 1983] :



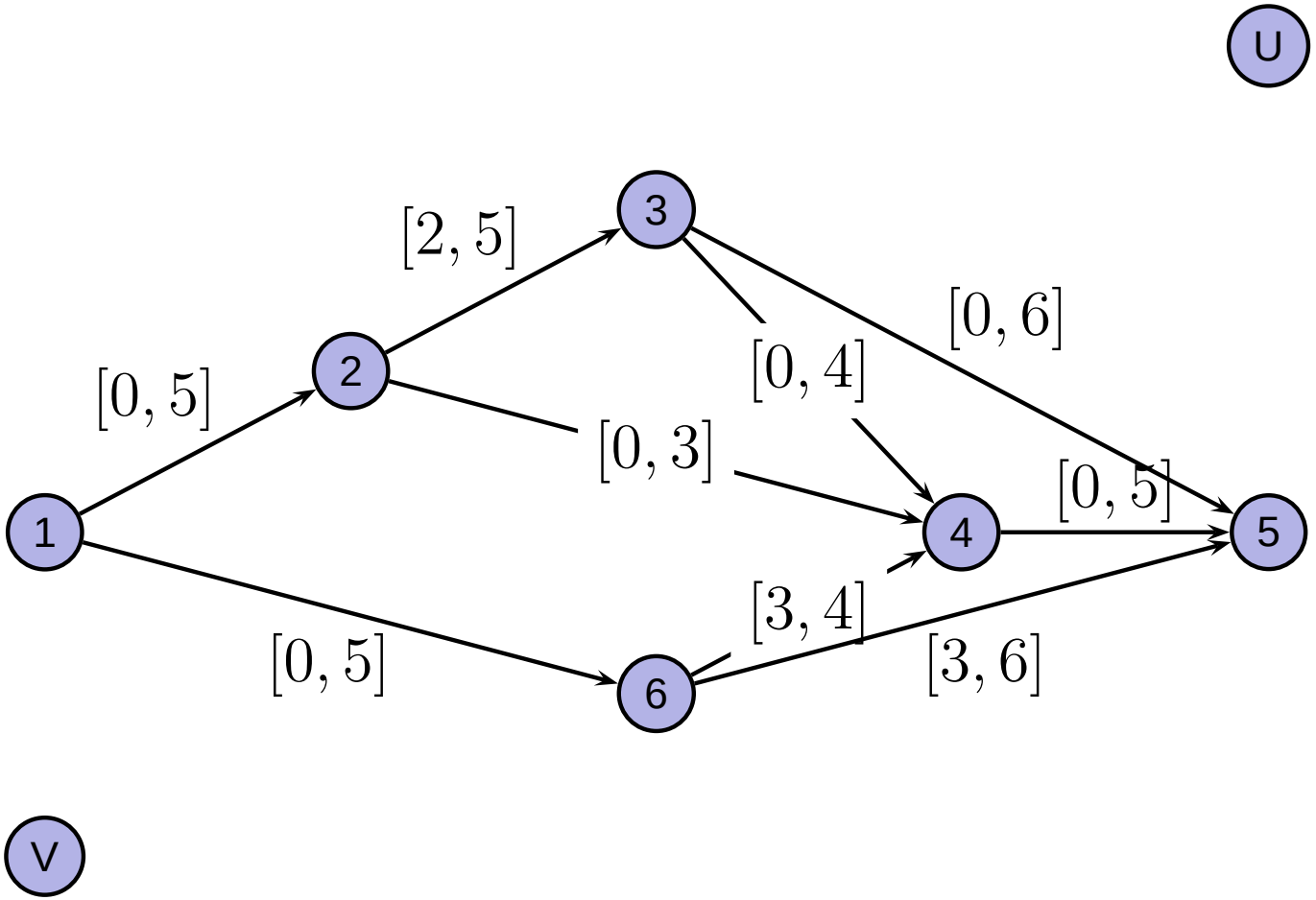
- En cas d’échec, on obtient une inégalité

$$\sum_C B^-(i) + \sum_C U'ij + \sum_C B^+(i) < \sum_i B^+(i)$$

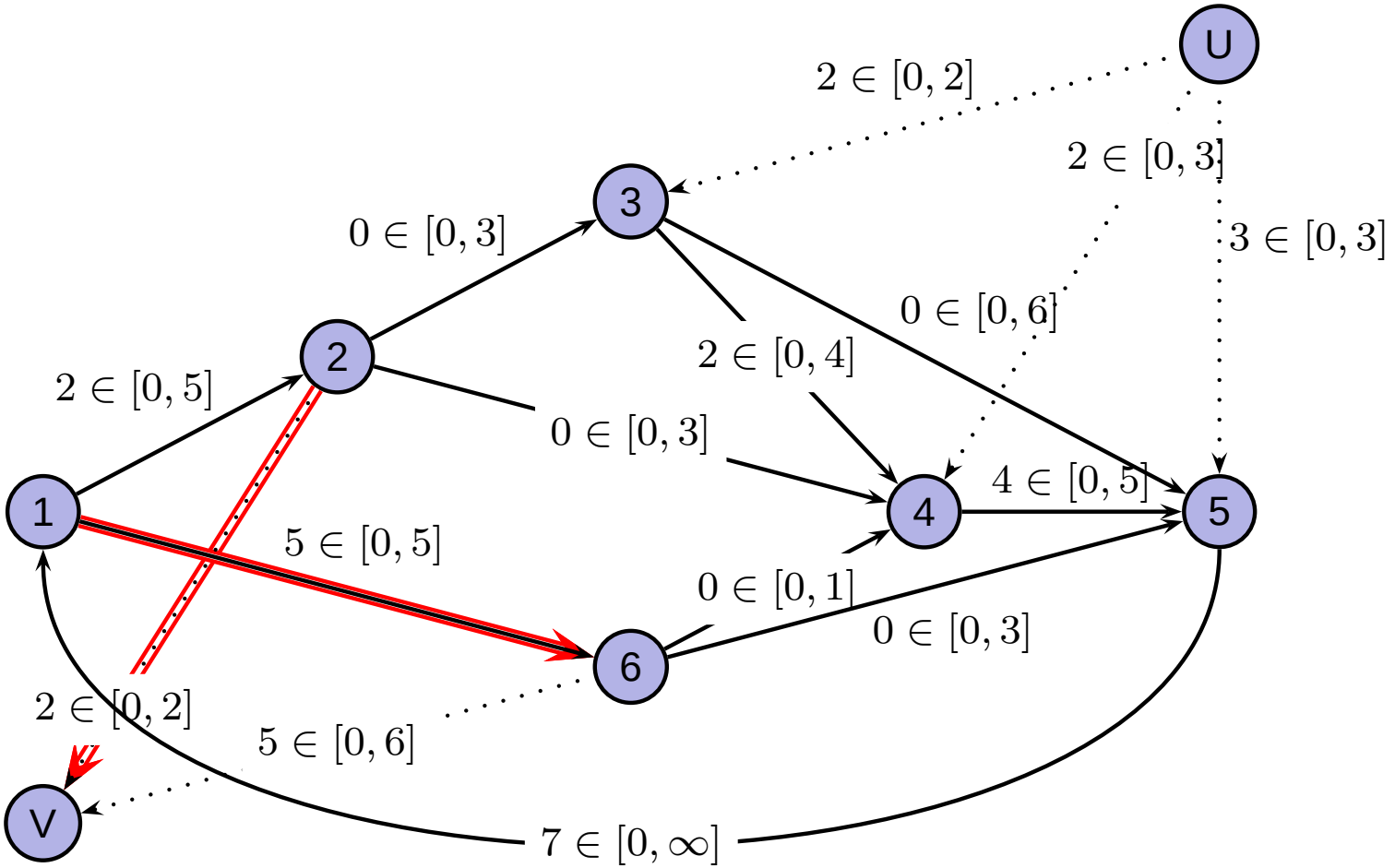
- D’où l’explication de contradiction suivante :

$$\bigcup (expl(bsup(U_{ij})) \cup expl(binf(U_{ij}))) \cup \bigcup expl(binf(L_{ik}))$$

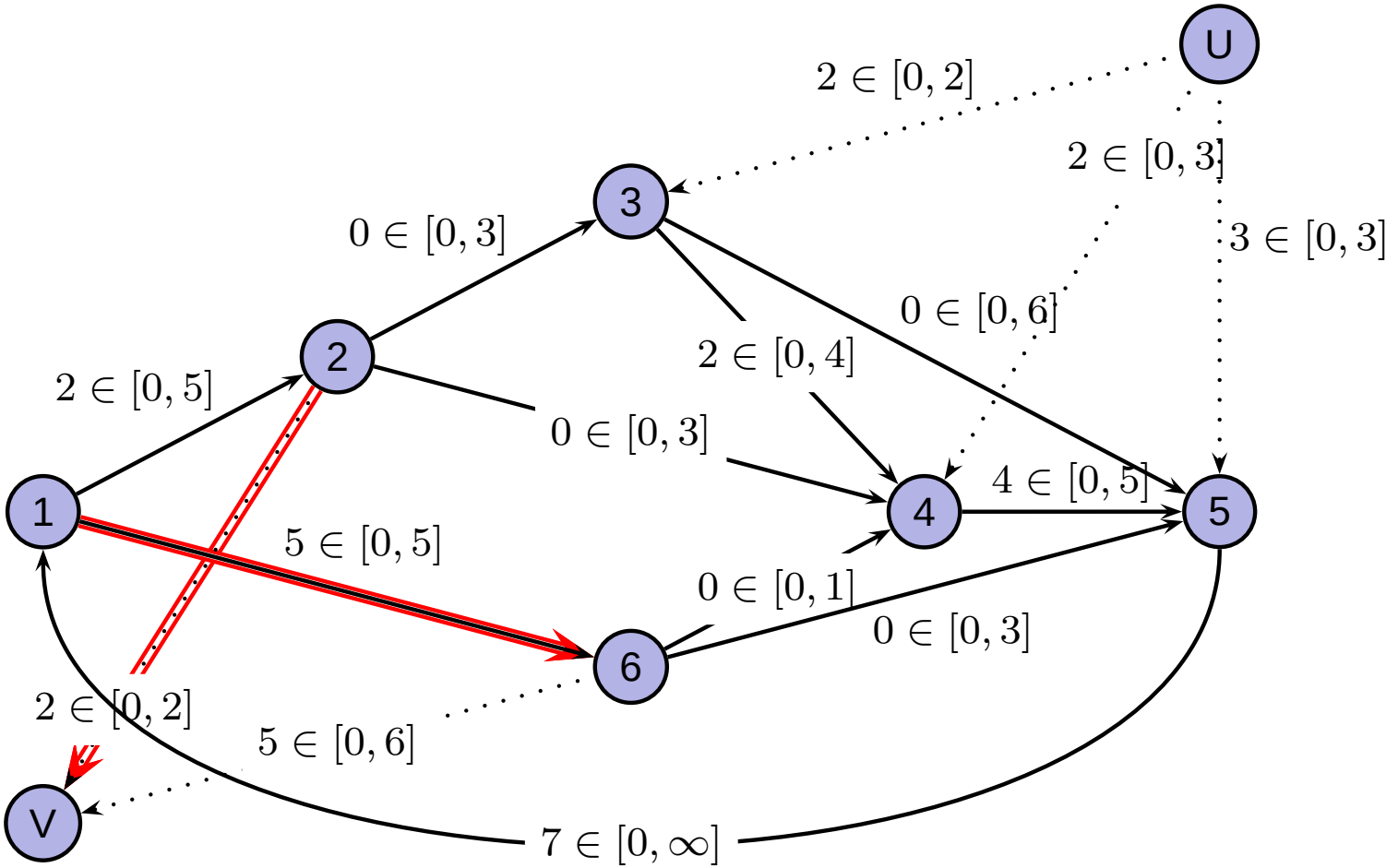
Exemple



Exemple



Exemple



$$expl(cont.) = expl(bsup(V_{1,6})) \cup expl(binf(V_{1,6}))$$
$$\cup expl(binf(V_{6,4})) \cup expl(binf(V_{6,5}))$$

Quelques résultats

La contrainte a été testée en simulant une contrainte “Global Sequencing Constraint” par deux flots :

- les explications semblent être confirmées expérimentalement,

Quelques résultats

La contrainte a été testée en simulant une contrainte “Global Sequencing Constraint” par deux flots :

- les explications semblent être confirmées expérimentalement,
- des gains en nombre de backtracks sont obtenus,

Quelques résultats

La contrainte a été testée en simulant une contrainte “Global Sequencing Constraint” par deux flots :

- les explications semblent être confirmées expérimentalement,
- des gains en nombre de backtracks sont obtenus,
- une version efficace en cours d’implémentation

Conclusion et perspectives

- Les algorithmes fournis par la R.O. nous a permis d’“expliquer” la contrainte de flot.
- Ces résultats pourraient être utilisés pour d’autres contraintes s’appuyant sur le flot.
- Les résultats expérimentaux semblent encourageant.

- Ces techniques pourraient être généralisées :
  - Cadre de [Beldiceanu, 2000],
  - Cadre distribué,
  - Coopération entre solveurs. . .

ECOLE DES MINES DE NANTES

Des questions ?

Références

[Beldiceanu, 2000] Beldiceanu, N. (2000). Global constraints as graph properties on a structured network of elementary constraints of the same type. In *Principles and Practice of Constraint Programming*, pages 52–66.

[Benoist et al., 2002] Benoist, T., Gaudin, E., and Rottembourg, B. (2002). Constraint Programming Contribution to Benders Decomposition : A Case Study. In *Principles and Practice of Constraint Programming*.

[Berge, 1983] Berge, C. (1983). *Graphes*. Gauthiers-villars, Paris, 3e édition.

[Rochart and Jussien, 2003] Rochart, G. and Jussien, N. (2003). Une contrainte `stretch` expliquée. In *9ièmes Journées nationales sur la résolution pratique de problèmes NP-complets (JNPC’03)*.