Modélisation et Contraintes Globales en Programmation par Contraintes

Jean-Charles REGIN
ILOG, Sophia Antipolis
regin@ilog.fr

Plan

- Principes de la Programmation par Contraintes et problématique de la modélisation
- Algorithme de filtrage et contraintes globales
- Problèmes sur-contraints
- Applications
- Un exemple détaillé : la contrainte de séquence
- Perspectives
- Conclusion

Programmation par Contraintes (PPC)

- 3 notions:
  - réseau de contraintes : variables, domaines, contraintes + filtrages (réduction de domaines)
  - propagation
  - procédure de recherche (affectations + backtracks (retour arrière))

Problème = conjonction de sous-problèmes

- En PPC un problème peut être vu comme une conjonction de sous-problèmes que l'on est capable de résoudre
- Un sous-problème peut être trivial : $x < y$ ou complexe : recherche d'un flot compatible.
- Un sous-problème = une contrainte

Contraintes

- Contraintes prédéfinies : arithmétiques ($x < y$, $x = y + z$, $|x-y| > k$), alldiff, cardinalité, séquence
- Contraintes données en extension par la liste des combinaisons de valeurs autorisées ou interdites
- Contraintes définies par l'utilisateur : n'importe quel algorithme peut être intégré
- Contraintes obtenues par combinaison logique de contraintes en utilisant les opérateurs OR, AND, NOT, XOR. Parfois appelées méta-contraintes.

Filtrage

- Etre capable de résoudre un sous-problème signifie que l'on dispose d'une méthode pour le résoudre
- La PPC utilise cette méthode pour supprimer des valeurs qui ne peuvent pas appartenir à une solution de ce sous-problème. C'est le **filtrage ou la réduction de domaines.**
- E.g: x < y et D(x)=[10,20], D(y)=[5,15] => D(x)=[10,14], D(y)=[11,15]

Filtrage

- Un algorithme de filtrage est associé à chaque contrainte (sous-problème).
- Peut être simple ($x < y$) ou complexe (alldiff)

Consistance d'arc

- Toutes les valeurs qui n'appartiennent pas à une solution sont supprimées.
- Exemple: Alldiff({x,y,z}) with
  D(x)=D(y)={0,1}, D(z)={0,1,2}
  les deux variables x et y ne peuvent prendre que les deux valeurs 0 et 1, aussi z ne peut pas prendre une de ces valeurs.
- FA par AC => 0 et 1 sont supprimées de D(z)

Propagation

- Les réductions de domaines dûes à une contrainte peuvent conduire à de nouvelles réductions de domaines pour d'autres variables.
- Lorsqu'un domaine est modifié toutes les contraintes impliquant cette variable sont étudiées et ainsi de suite…

Propagation

- D(x)=D(y)={0,1}, D(z)={0,1,2}, D(u)={3,4,5}, D(v)={4,5,6}
- Alldiff({x,y,z}), u=z+3, v>u, |z-v|<6

Propagation

- D(x)=D(y)={0,1}, D(z)={0,1,2}, D(u)={3,4,5}, D(v)={4,5,6}
- Alldiff({x,y,z}), u=z+3, v>u, |z-v|<6
- Alldiff({x,y,z}): FA=> z=2

Propagation

- D(x)=D(y)={0,1}, D(z)={0,1,2}, D(u)={3,4,5}, D(v)={4,5,6}
- Alldiff({x,y,z}), u=z+3, v>u, |z-v|<6
- Alldiff({x,y,z}): FA=> z=2
- u=z+3: FA=> u=5

Propagation

- D(x)=D(y)={0,1}, D(z)={0,1,2}, D(u)={3,4,5}, D(v)={4,5,6}
- Alldiff({x,y,z}), u=z+3, v>u, |z-v|<6
- Alldiff({x,y,z}): FA=> z=2
- u=z+3: FA=> u=5
- v>u: FA=> v=6

Propagation

- D(x)=D(y)={0,1}, D(z)={0,1,2}, D(u)={3,4,5}, D(v)={4,5,6}
- Alldiff({x,y,z}), u=z+3, v>u, |z-v|<6
- Alldiff({x,y,z}): FA=> z=2
- u=z+3: FA=> u=5
- v>u: FA=> v=6
- |z-v|<6: FA => y=1

Propagation

- D(x)=D(y)={0,1}, D(z)={0,1,2}, D(u)={3,4,5}, D(v)={4,5,6}
- Alldiff({x,y,z}), u=z+3, v>u, |z-v|<6
- Alldiff({x,y,z}): FA=> z=2
- u=z+3: FA=> u=5
- v>u: FA=> v=6
- |z-v|<6: FA => y=1
- Alldiff({x,y,z}): FA=> x=0
- Solution: x=0, y=1, z=2, u=5, v=6

Pourquoi utiliser la Propagation?

- Un problème = conjonction de sous-problèmes faciles.
- Un sous-problème offre un point de vue local. La propagation essaie d'obtenir un point de vue global à partir d'un ensemble indépendant de point de vues locaux.
- La conjonction est plus forte que l'union de résolutions indépendantes.

Comment aider la Propagation ?

- Pour aider la propagation à avoir un point de vue global on utilise des **contraintes globales** !
- Une contrainte globale = conjonction de contraintes

Contraintes globales

- D(x1)=D(x2)=D(x3)=D(x4)={a,b,c,d}
- Contraintes : b,c et d doivent être prises au moins 1 fois.
- Algorithme de Filtrage : si b n'est pas affectée et s'il y a une seule variable x qui contient b dans son domaine alors x=b
- Problème : si x1=a et x2=a alors rien ne peut être déduit, alors qu'il n'y a pas de solution
- De la présence simultanée de contraintes on peut déduire plus de choses : a peut être prise au plus 1 fois et b,c,d peuvent être prise au plus 2 fois.

Contraintes globales

- Une contrainte globale est la conjonction de contraintes. Cela permet souvent de déduire des contraintes implicites à partir de la présence simultanée de contraintes
- C'est le cas pour l'exemple précédent avec la contrainte globale de cardinalité

Contrainte globale

- Si C est la conjonction de C1, C2, … Ck alors la consistance de C est equivalente à la consistance du réseau de contraintes définies par les l'union des contraintes C1, C2, … Ck
- Idem pour la consistance d'arc

Contrainte globale : alldiff

- Une contrainte alldiff définie sur un ensemble de variables X impose que les variables doivent prendre des valeurs deux à deux différentes
- Elle correspond à un nombre quadratique de contraintes ≠

Contrainte globale : alldiff

- Colorier le graphe avec les cliques :
  c0 = {0, 1, 2, 3, 4}
  c1 = {0, 5, 6, 7, 8}
  c2 = {1, 5, 9, 10, 11}
  c3 = {2, 6, 9, 12, 13}
  c4 = {3, 7, 10, 12, 14}
  c5 = {4, 8, 11, 13, 14}
- clique taille: 27 — Global: #fails: 0 cpu time: 1.212 s; Local: #fails: 1 cpu time: 0.171 s
  clique taille : 31 — Global: #fails: 4 cpu time: 2.263 s; Local: #fails: 65 cpu time: 0.37 s
  clique taille : 51 — Global: #fails: 501 cpu time: 25.947 s; Local: #fails: 24512 cpu time: 66.485 s
  clique taille : 61 — Global: #fails: 5 cpu time: 58.223 s; Local: ????????????

Recherche de solutions

- Algorithme de Backtrack avec stratégies:
  on essaie d'affecter successivement des valeurs aux variables. Si un échec se produit alors on backtracke et on essaie une autre valeur pour la variable.
- Stratégie : définir quelle variable et quelle valeur vont être affectées ensemble.
- Après chaque réduction de domaines, donc après chaque affectation, les filtrages et la propagation sont déclenchés.

Plan

- Principes de la Programmation par Contraintes et problématique de la modélisation
- **Algorithme de filtrage et contraintes globales**
- Problèmes sur-contraints
- Applications
- Un exemple détaillé : la contrainte de séquence
- Perspectives
- Conclusion

Algorithmes de filtrage et contraintes globales

- Algorithmes génériques
- Intégration d'algorithmes de Recherche Opérationnelle
- Contraintes dérivées
- Autres contraintes globales

Algorithmes génériques

- Algorithmes dans le cas binaires (CAC)
- Algorithmes dans le cas non binaires (GAC-Schema)

Algorithmes générique dans le cas binaire

- La contrainte est donnée en extension par des paires de valeurs compatibles. But : calculer l'AC le plus vite possible
- AC-3 [Mackworth 77],
  AC-4 [Mohr et Henderson 86],
  AC-6 [Bessiere et Cordier 93],
  AC-7 [Bessiere et Regin 94],
  AC-Inference, AC-Identical, [Bessiere, Freuder et Regin 99],
  AC-2001 [Bessiere et Regin 01],
  CAC [Regin 04].

GAC-Schema: instanciation

- Liste des tuples autorisés
- Liste des tuples interdits
- Prédicats
- N'importe quel algorithme de RO
- Réentrance du solver
- [Bessiere et Regin 97], [Bessiere et Regin 99]

GAC-Schema

- Idée :

  tuple = solution d'une contrainte

  support = tuple valide

  - tant que le tuple est valide : ne rien faire
  - si le tuple n'est plus valide, alors rechercher un nouveau support pour les valeur supportées par le tuple

- une solution (support) peut être calculée par n'importe quel algorithme de RO. Il faut une solution et pas seulement prouver qu'elle existe.

GAC-Schema: complexité

- CC complexité pour vérifier la consistance (recherche dans une table, appel d'un algorithme de RO) : recherche pour un support coûte CC
- n variables, d valeurs:
  pour chaque valeur : CC
  pour toutes les valeurs : O(ndCC)
- **A partir d'un algorithme de RO** calculant une solution, **la consistance d'arc** peut être calculé en **O(ndCC).**

Pré-résolution d'une partie d'un problème GAC-Schema + autorisés

- Problème de Configuration :

  5 types of components: {glass, plastic, steel, wood, copper}
  3 types of bins: {red, blue, green} whose capacity is red 5, blue 5, green 6
  Constraints:
  - red can contain glass, cooper, wood
  - blue can contain glass, steel, cooper
  - green can contain plastic, copper, wood
  - wood require plastic; glass exclusive copper
  - red contains at most 1 of wood
  - green contains at most 2 of wood

  For all the bins there is either no plastic or at least 2 plastic
  Given an initial supply of 12 of glass, 10 of plastic, 8 of steel, 12 of wood and 8 of copper; what is the minimum total number of bins?

Pré-résolution d'une partie d'un problème GAC-Schema + autorisés

	#bk	temps
modèle standard	1,361,709	430
GAC+autorisés	12,659	9.7

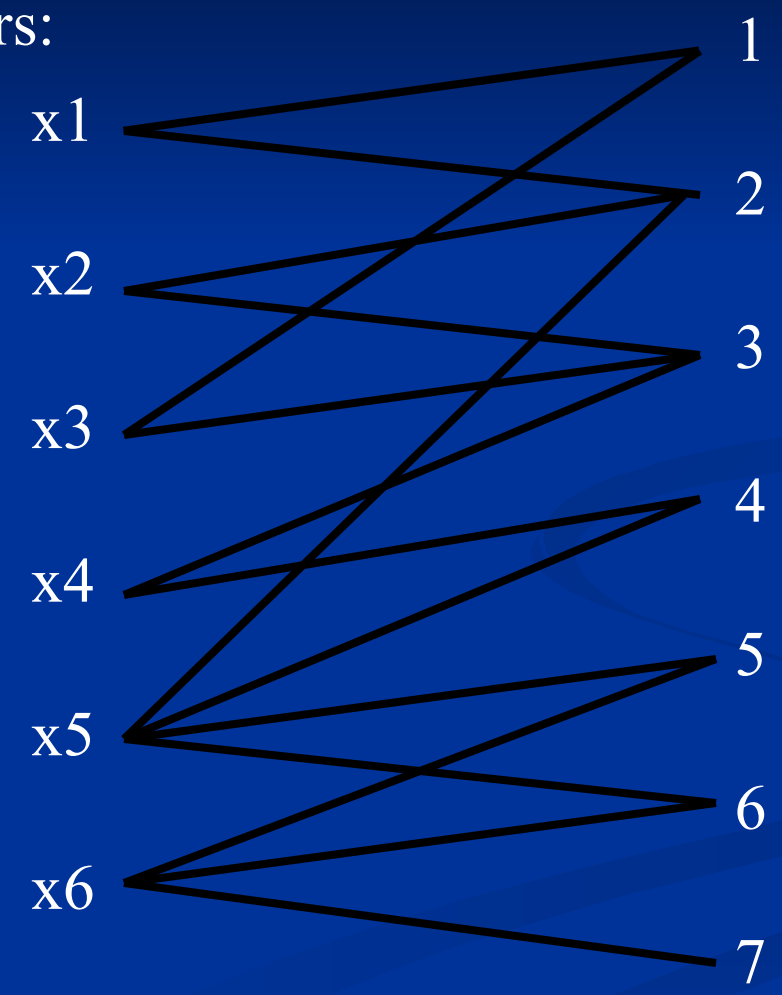
Intégration d'algorithmes de Recherche Opérationnelle

- Alldiff et couplage biparti [Regin 94]
- Symmetric alldiff et couplage [Regin 99]
- Contrainte globale de cardinalité et flot [Regin 96]
- Contrainte globale de cardinalité avec coûts et flot à coût minimum [Regin 99]

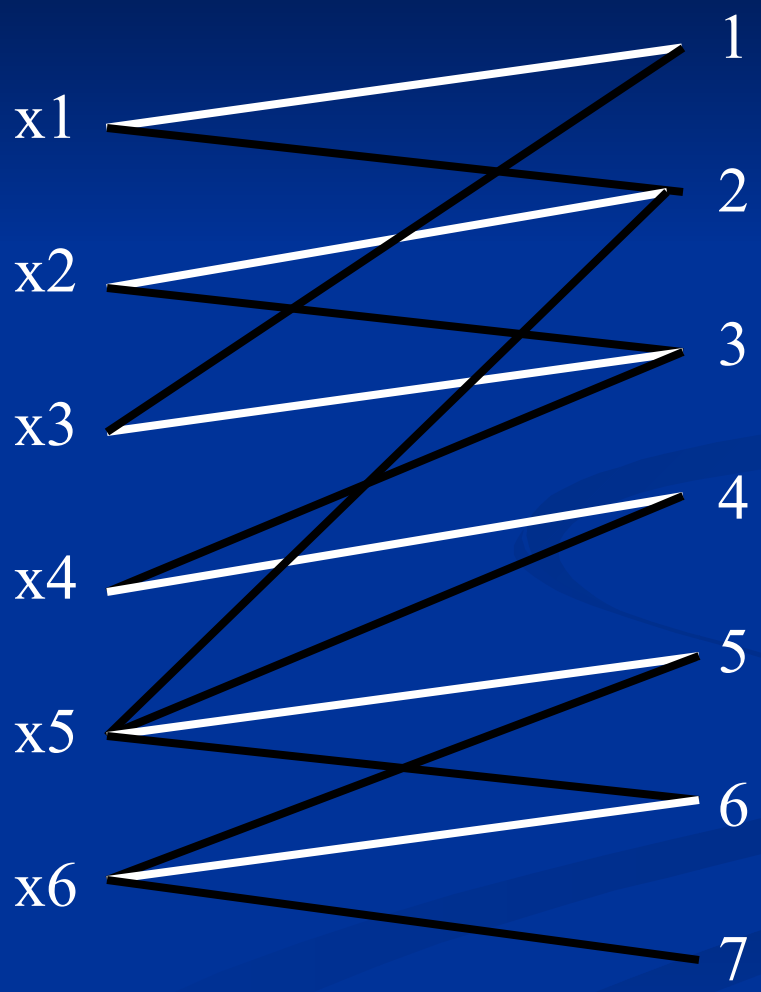
Contrainte Alldiff

Graphe des valeurs:

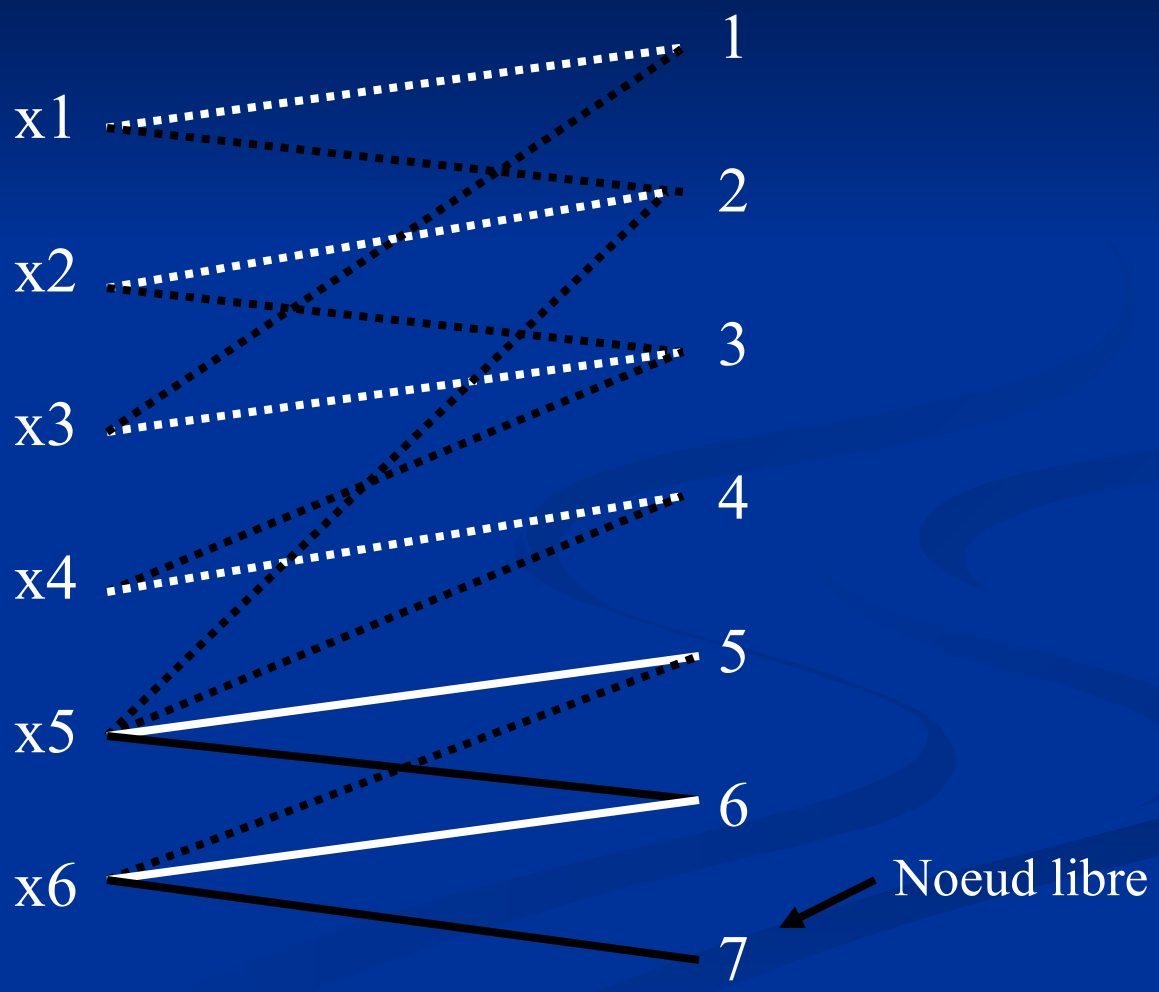
D(x1)={1,2}
D(x2)={2,3}
D(x3)={1,3}
D(x4)={3,4}
D(x5)={2,4,5,6}
D(x6)={5,6,7}



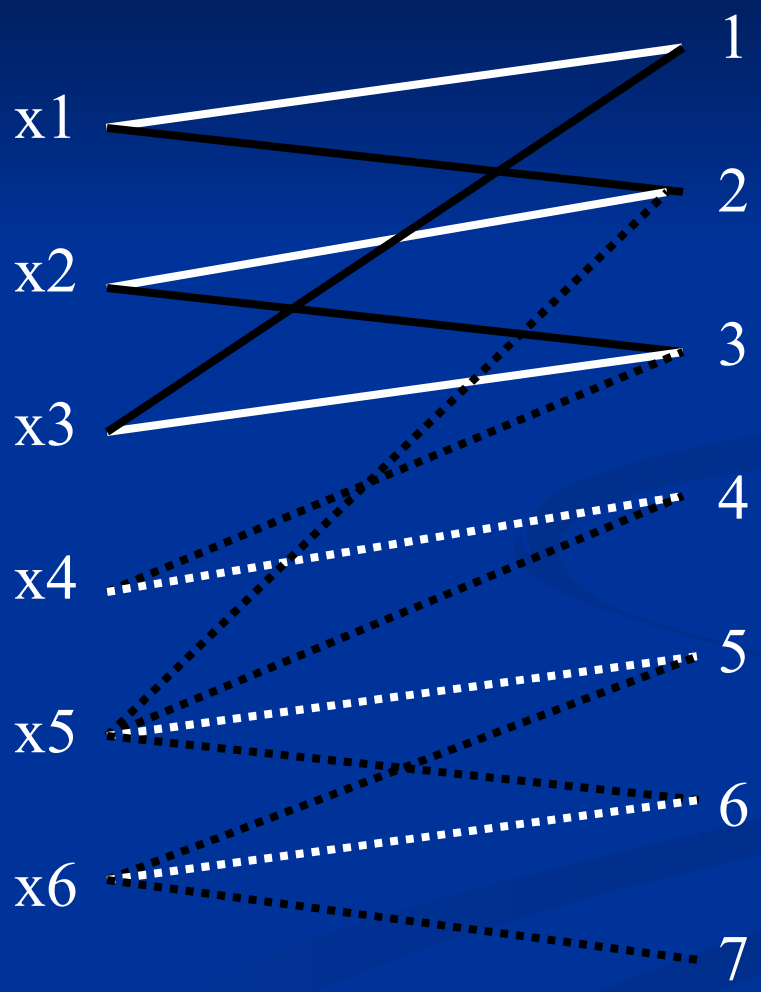
Couplage



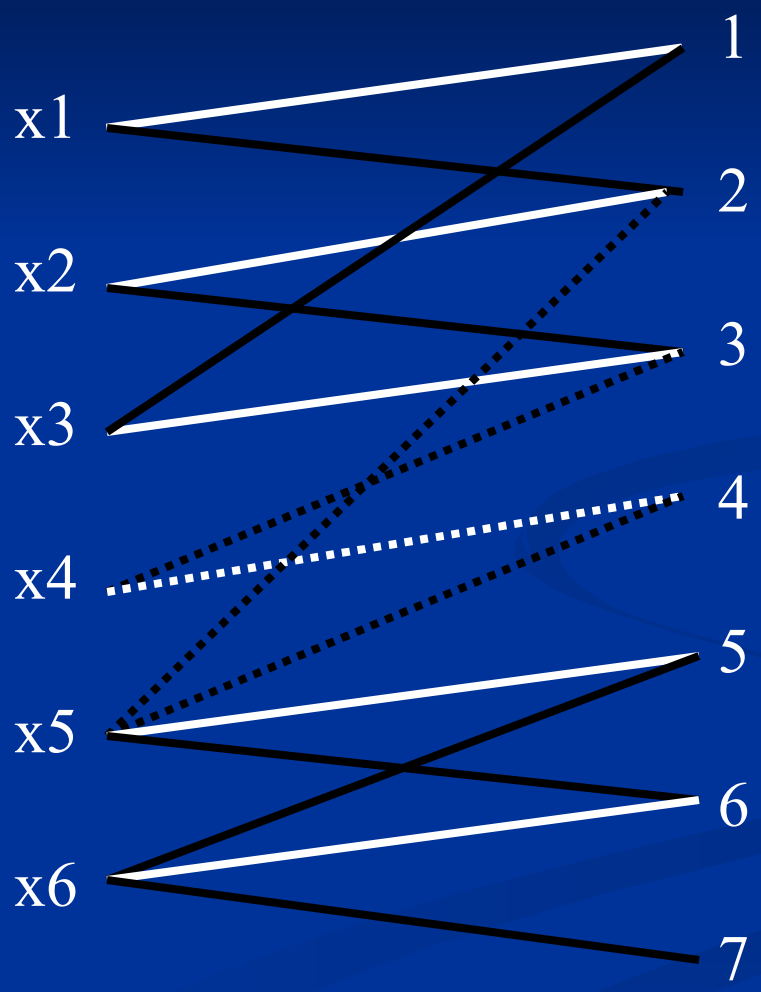
Chaîne alternée



Cycle alterné



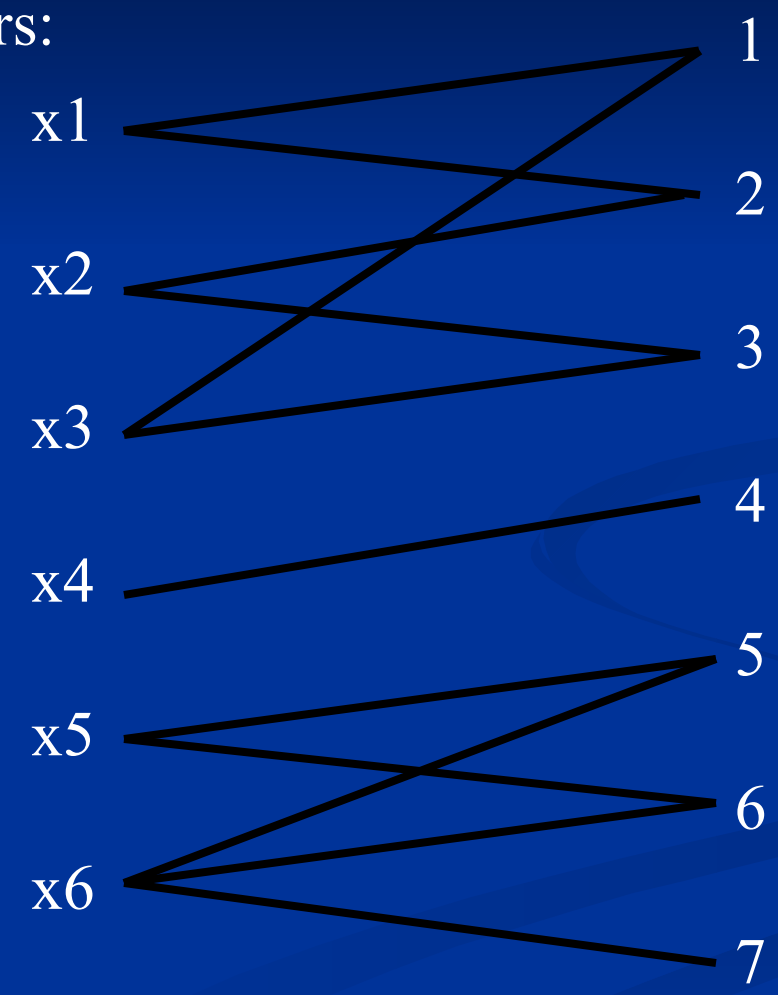
Contrainte Alldiff



Consistance d'arc

Graphe des valeurs:

D(x1)={1,2}
D(x2)={2,3}
D(x3)={1,3}
D(x4)={4}
D(x5)={5,6}
D(x6)={5,6,7}



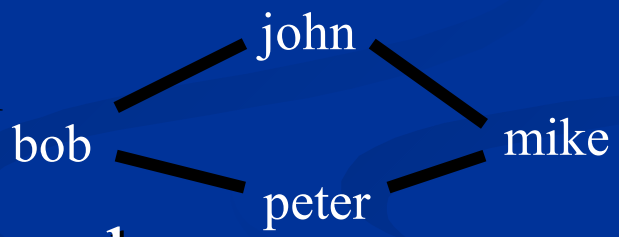
Alldiff

- Consistance = calcul d'un couplage
- AC en O(m)

Symmetric alldiff

- But : grouper des entités par paires
- Exemple: pilotes d'avions, infirmières…

  Liste de compatibilité
  bob peut travailler avec john et peter
  john peut travailler avec bob et mike
  mike peut travailler avec peter et john
  peter peut travailler avec bob et mike

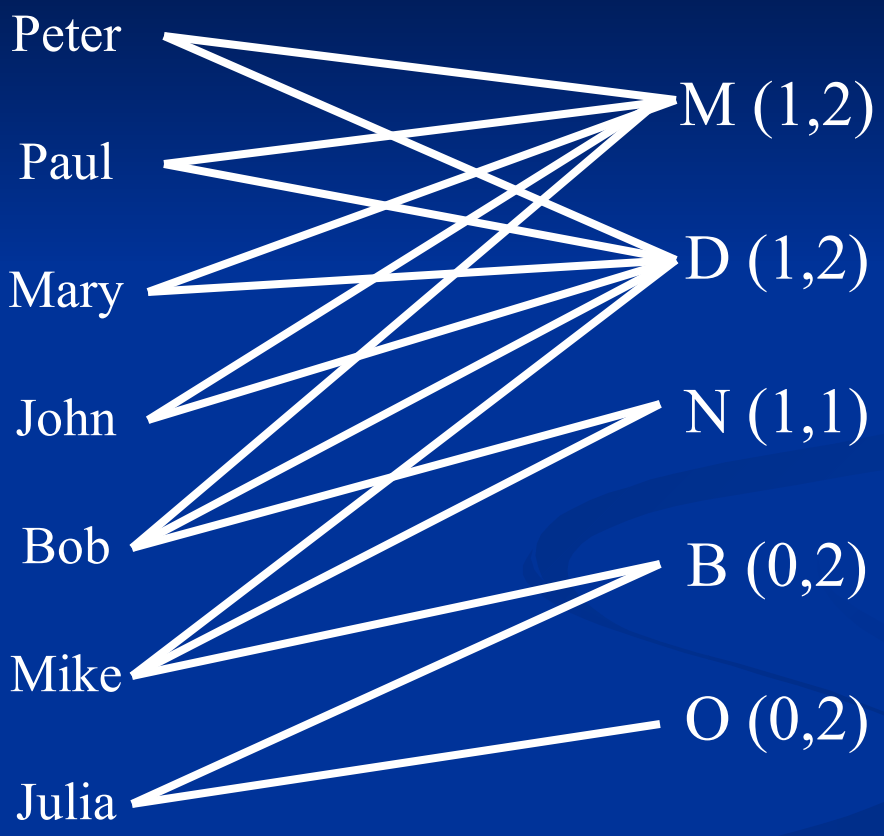


- Consistance = calcul d'un couplage
- AC = nO(m)
- Autre filtrage proposé

Contrainte Globale de Cardinalité

- GCC(X,{l_i},{u_i})
- Définie sur un ensemble X de variables
- Le nombre de fois où chaque valeur v_i peut être prise doit appartenir à un intervalle donné [l_i, u_i]
- Exemple: D(x1)={a,b,c,d}, D(x2)={a,b,c,d}, D(x3)={b,c},D(x4)={c,d}. Les valeurs b et c doivent être prises au plus 2 fois, les valeurs a et d doivent être prises au moins une fois.

GCC



GCC

- Consistance = calcul d'un flot compatible dans un réseau biparti
- AC = O(m)

GCC + coûts

- On ajoute des coûts sur les arcs et on contraint la somme des coûts à être inférieur (ou supérieur) à une valeur donnée
- Consistance = calcul d'un flot à coût minimum
- AC = n fois un plus court chemin

Contraintes dérivées

- Alldiff avec coûts [Regin 99]
- Contrainte de plus court chemin
- Somme et produit scalaire de variables toutes différentes [Regin 99]
- Contrainte k-diff [Regin 95]
- Contrainte sur des variables ensemblistes

Autres contraintes globales

- Contrainte combinant une somme et des inégalités binaires [Regin et Rueher 00]
- Contrainte globale de séquence [Regin et Puget 97]

Plan

- Principes de la Programmation par Contraintes
- Problématique de la modélisation
- Algorithme de filtrage et contraintes globales
- **Problèmes sur-contraints**
- Applications
- Un exemple détaillé : la contrainte de sequence
- Perspectives
- Conclusion

Problèmes sur-contraints

- Aucune solution ne satisfait toutes les contraintes
- Que pouvons-nous faire ?
- Certaines contraintes doivent être relachées
  - Contraintes dures : doivent être satisfaites
  - Contraintes molles : peuvent être violées

Problèmes sur-contraints
Plan

- Contraintes molles et algorithmes de filtrage
  - Comment relâcher une contrainte ?
  - Comment utiliser la structure d'une contrainte molle ?
- Comment modéliser les problèmes sur-contraints
  - Constraintes sur les violations
  - Meta-contraintes
- Contraintes globales molles
- Discussion

Contraintes molles et algorithmes de filtrage

- **Sans autre information** : accepter la violation d'une constrainte signifie en fait que l'on considère que toute combinaison de valeurs satisfait la contrainte
- Cela signifie que la contrainte devient une contrainte universelles et donc **que l'on perd toute structure**
- D'où problème avec les algorithmes de filtrage (FA) :
  - FA exploitent la structure des contraintes
  - FA sont inefficaces quand tout est possible

Contrainte molle

- La violation d'une contrainte est associée à un coût qui peut être :
  - Le même pour toutes les violations
  - Dépendre de la violation
- Exemple: $x < y$, si $x \geq y$ on peut avoir :
  - Un coût fixe : cost = c
  - Un coût dépendant de la violation :
    cost = x –y ou cost = $(x-y)^2$

Utilisation de la structure de la violation : $x \leq y$

- [Regin, Petit, Bessiere, Puget 00, 01]
- Structure
  - Si la contrainte est satisfaite alors *cost* = 0
  - Si la contrainte est violée *cost* = x - y
- Algorithme de Filtrage :
  - D(x) = [90000,100000], D(y) = [99990,200000]
  - On déduit immédiatement que max(cost)= max(x) - min(y) = 10

Comment relâcher une contrainte ?

- Essayer de conserver une certaine structure à la contrainte dans le but de disposer d'algorithmes de filtrage puissants
- Définir une variable de coût et intégrer cette variable dans une contrainte

Problèmes sur-contraints
Plan

- Contraintes molles et algorithmes de filtrage
  - Comment relâcher une contrainte ?
  - Comment utiliser la structure d'une contrainte molle ?
- **Comment modéliser les problèmes sur-contraints**
  - **Constraintes sur les violations**
  - **Meta-contraintes**
- Contraintes globales molles
- Discussion

Contraintes sur les violations

- Dans les problèmes réels, le but est plus complexe que la minimisation du nombre des contraintes violées. [Petit, Regin, Bessiere 00]

⇒ Règles sur les violations

- Distinction entre les contraintes dures et molles
- Priorités
- Contrôle de la distribution des violations dans le réseau de contraintes : une répartition homogène des violations est généralement requise
- Dépendances spécifique entre les contraintes

Contraints sur les violations : Priorités

- Toutes les contraintes n'ont pas la même importance.
- But : favoriser la satisfaction des contraintes les plus importantes

(C1: contraint dure)
C2: cruciale
C3: importante
C4: importance faible
C5: préférence

Répartition homogène des violations

- Dans de nombreuses applications réelles les violations doivent être réparties de façon homogène
- Des règles plus complexes peuvent être imposées sur la distribution des violations.

Répartition homogène des violations

- Exemple
  - travailleur W_1: préférences = { C_{11}, C_{12}, C_{13}, ... }
  - travailleur W_2: préférences = { C_{21}, C_{22}, C_{23}, ... }
  - travailleur W_3: préférences = { C_{31}, C_{32}, C_{33}, ... }
- Un emploi du temps tel que certains travailleurs ont toutes leurs préférences satisfaites et d'autres aucune n'est pas acceptable.
- Pour chaque Wi:
  - Au moins j contraintes satisfaites
  - Au moins j contraintes satisfaites, et au moins k contraintes violées avec degré < m, etc.
- Idée générale : éviter d'avoir des périodes de travail très difficle et ensuite de n'avoir presque rien à faire. Vrai pour des personnes ou des machines.

Contraintes sur les violations : Dépendances

- Exemple
  - « si C_1 et C_2 sont violées alors C_3 doit être satisfaite »

Méta Contrainte

- [Regin, Petit, Bessiere 00]
- $s_k > 0$ exprime que Ci est violée (distance à la satisfaction)
- $s_k = 0$ exprime que Ci est satisfaite
- $D(s_k)$ est un domaine entier
- Chaque contrainte molle est remplacée par la disjonction :

$$[\ (s = 0) \wedge C\ ] \ \vee \ [\ (s > 0) \wedge \neg C\ ]$$

Méta Contrainte

- Comme les valuations sont exprimées au travers de variables, les contraintes sur ces variables peuvent être ajoutées pour exprimer des règles globales sur les violations

Max-SAT = Contrainte de Somme de Satisfaction

- Dans la contrainte *ssc*, chaque contrainte Ci est remplacée par :

$$[\ (C_i \wedge (u_i = 0)) \ \vee (\neg C_i \wedge (u_i = 1))\ ]$$

- Une variable *unsat* est utilisée pour exprimer l'objectif :

$$[\ unsat\ = \sum_{i=1}^{\#\,Ci} u_i\ ]$$

Avantages de ce modèle

- Problème d'optimisation classique sous contraintes
  - Intégration directe dans un solver
  - N'importe quel algorithme de recherche peut être utilisé, donc pas seulement un Branch and Bound.
- Lorsqu'une valeur est affectée à $u_i \in U$, l'algorithme de filtrage associé à C_i (resp. $\neg C_i$) peut être utilisé
- Aucune hypothèse n'est faite sur les contraintes.

Avantages de ce modèle

- Intégration du coût à l'intérieur de contraintes
- Contraintes sur les violations peuvent être facilement définies
- Contraintes globales molles

Définition générale du coût de violations

- [Petit, Regin et Bessiere 01]
- Deux coûts généraux de violations :
  - Coût de violation basé sur les variables
  - Coût de violation basé sur le graphe primal

Coût de violation basé sur les variables

- Combien de contraintes doivent être supprimées pour satisfaire la contrainte ?
- Alldiff({x1,x2,x3,x4,x5})
  (a,a,a,b,b) cost = 3
  (a,a,a,a,b) cost = 3

Coût de violation basé sur le graphe primal

- Valide pour une contrainte globale correspondant à une conjonction de contraintes Nombre de contraintes de la conjonction qui sont violées.
- Alldiff({x1,x2,x3,x4,x5})
  (a,a,a,b,b) coût = triangle(a,a,a) + paire (b,b)
  = 3 + 2 =5
  (a,a,a,a,b) coût = quadrangle (a,a,a,a)
  = 6

Méta-contrainte pour exprimer l'homogénéité

- Exemple: problème d'emploi du temps
  - *n* travailleurs expriment des préférences
  - Pour chacun au moins *j* préférences doivent être satisfaites
- Pour chaque travailleur on contraint le nombre de variable s_k à 0

Méta contraintes vs autres modèles

- Les Valued CSP ne sont pas capable de prendre en compte la structure des contraintes
- Les Valued CSP ne contiennent pas de “back propagation” parce qu’il n’y a pas de variable de coût
- Les Valued CSP ne sont pas capables de gérer les contraintes sur les violations
- Si on introduit une nouvelle contrainte sur les violations alors il n’est pas nécessaire de définir une nouvelle classe de CSP.

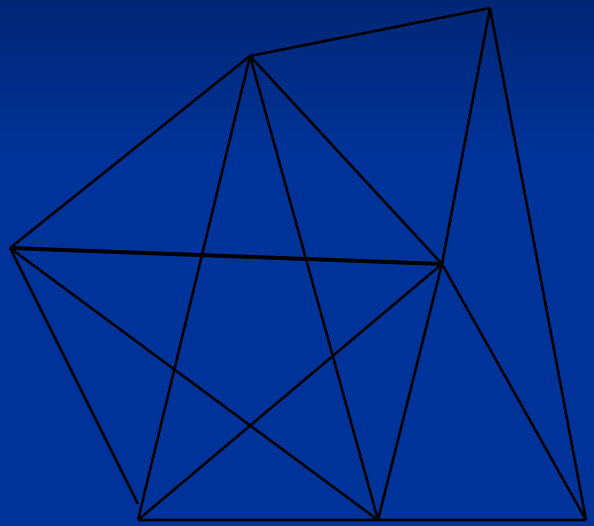
Plan

- Principes de la Programmation par Contraintes
- Problématique de la modélisation
- Algorithme de filtrage et contraintes globales
- Problèmes sur-contraints
- **Applications**
- Un exemple détaillé : la contrainte de sequence
- Perspectives
- Conclusion

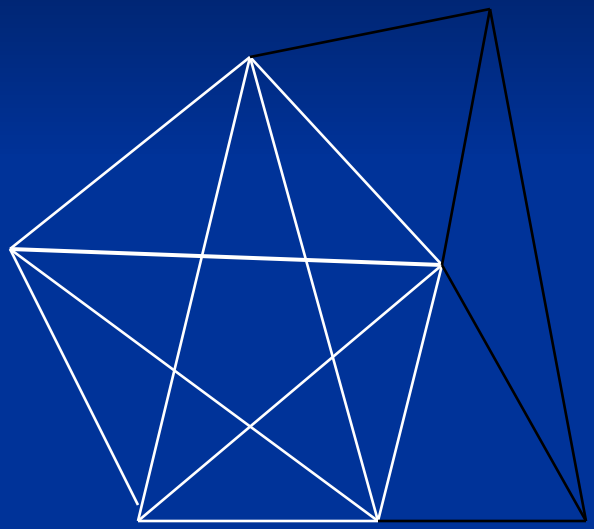
Max-Clique: le problème

- Etant donné un graphe G=(X,E)
- Une Clique est un ensemble de noeuds V de G tel que toute paire de noeuds de V est reliée par un arc
- MAX-CLIQUE : trouver la clique dont la cardinalité est maximum (la plus grande)

Maximum clique



Maximum clique



Motivations

- Contredire certaines croyances communes à propos de la PPC : ce n'est pas une bonne méthode pour résoudre les problème d'optimisation combinatoire et la PPC a besoin de temps pour trouver une bonne solution.
- Considérons un problème très connu d'optimisation combinatoire et montrons que la PPC
  - Est une technique efficace
  - Peut donner de bons résultats très rapidement

Le problème

- Advantage de ce problème : 400 références dans les états de l'art.
- Toutes les techniques existantes ont été étudiées (GA, Neural Network, Local Search, MIP …)
- DIMACS challenge en 1993.
- Domaine actif : papiers tous les ans

Résultats

- Tous les problèmes DIMACS avec moins de 400 noeuds sont résolus pour la première fois
- Idem pour 500 excepté pour un
- Un problème fermé en 150s

Résultats: PPC vs méthodes complètes

	Wood,	Ostegard,	Fahle,	Regin
#solved	38	36	45	52
< 10 min	38	35	38	44
best time	15	26	10	30
best LB	0	0	1	9

- Ostegard (programmation dynamique, RDS en PPC)
  en moins de 10 min: 350s, PPC: 285s

Résultats: PPC vs méthodes heuristiques

- Qualex: 50 meilleures bornes
- St-Louis, Gendron, Ferland : 50 meilleures bornes
- PPC : 58 (52 prouvées)
- PPC < 10 min: 49 (44 prouvées)
- PPC < 1 min: 41 (37 prouvées)

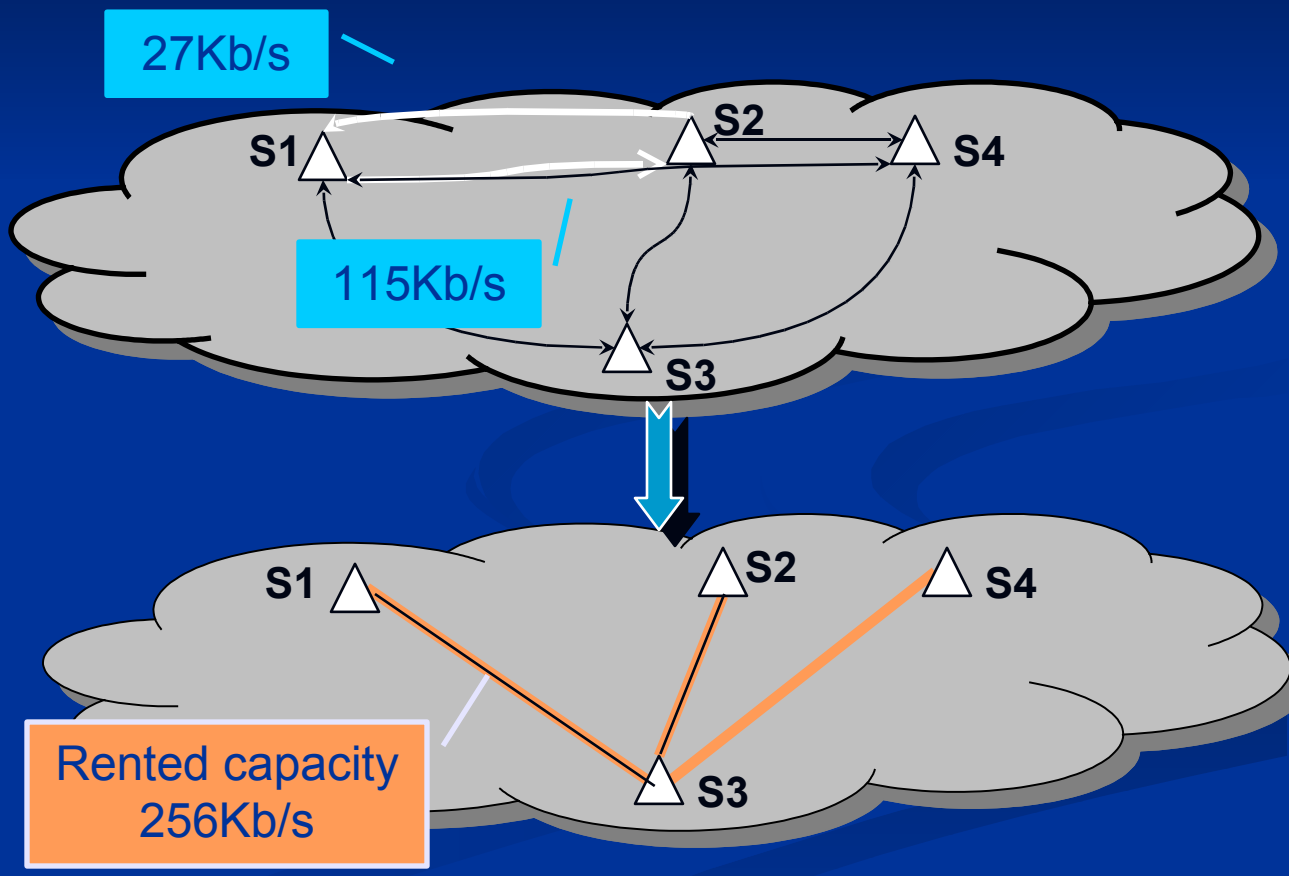
Le Problème ROCOCO (1)

Données :

- Traffic client
- Liens possibles, capacités et coûts

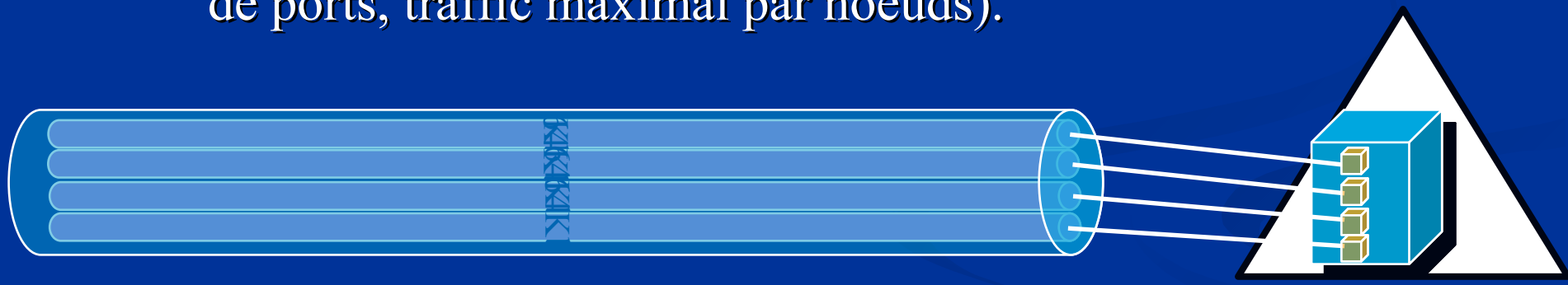
Résultat:

- **Définir les capacités de façon à satisfaire les demandes en minimisant le coût total**
- **Les demandes doivent êtres**



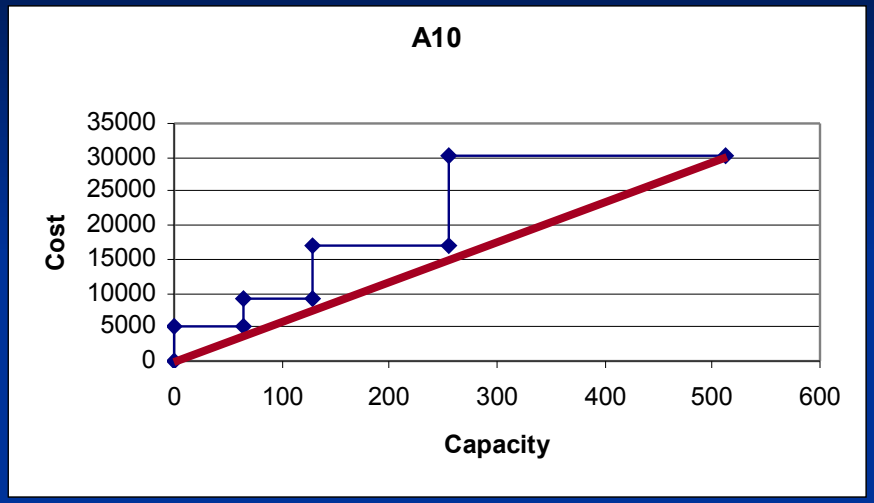
Le Problème ROCOCO (3)

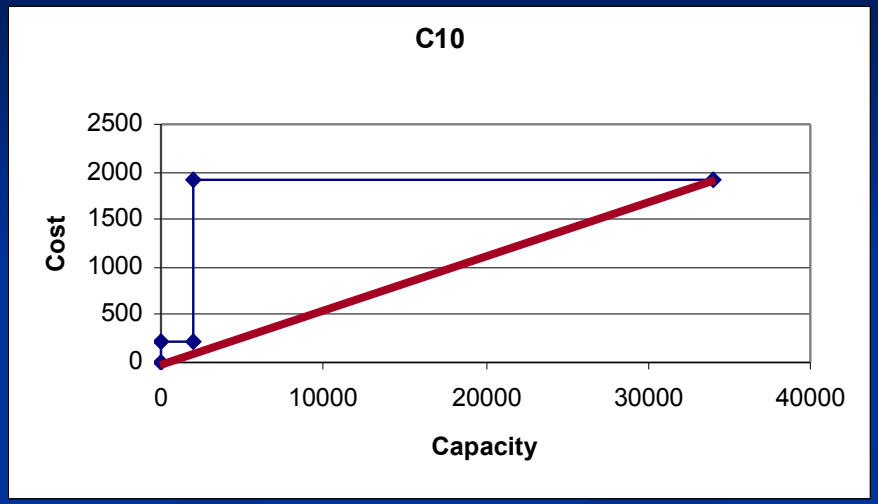
- Autres contraintes
  - Qualité de service
  - Réutilisation de l'équipement existant (limite sur le nombre de ports, traffic maximal par noeuds).

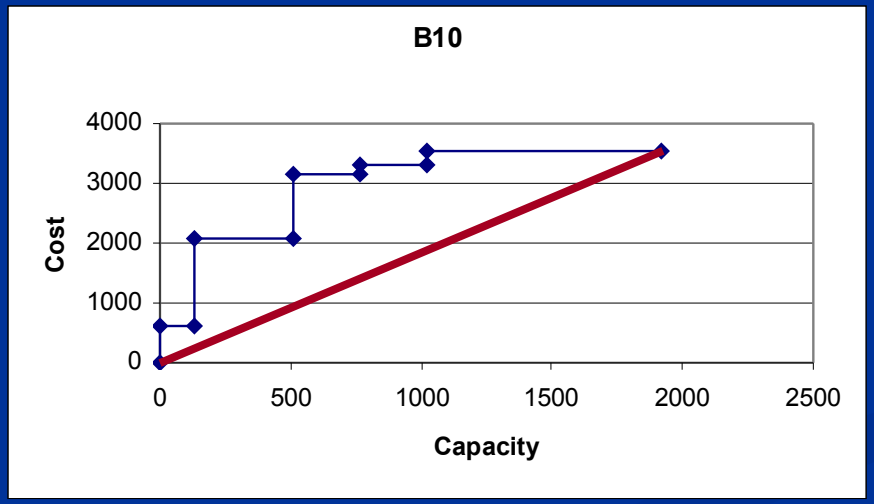


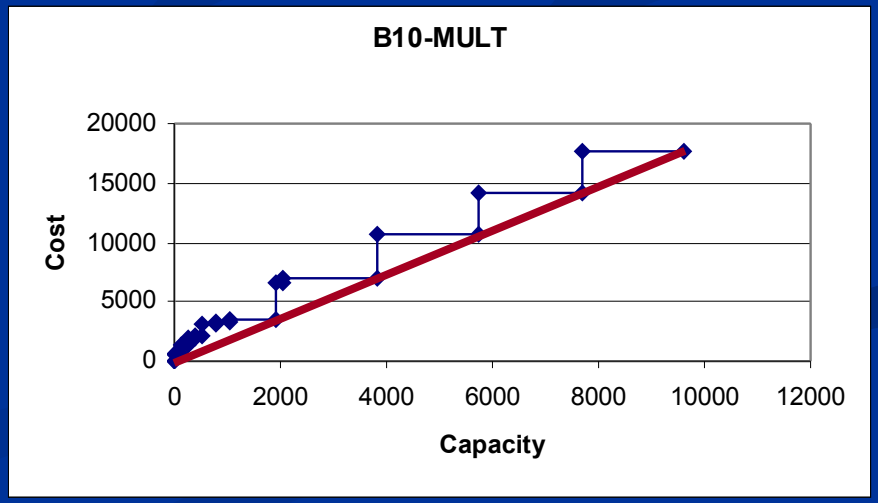
  - Contraintes commerciales and légales
  - Evolution possible future du réseau
  - Gestion de réseau (e.g., concentration du traffic)

Linéarisation









Solution en PPC

- [Le Pape, Perron, Regin et Shaw 02]
- Variables: chemins (Digraph Var)
  - Un ensemble d'arcs joignant l'origine de la demande à la destination
  - Fonctions de base : un arc (ou un noeud) est imposé ou interdit.
- Variables de dimensionnement : choix du niveau de capacité
- Contraintes spécifiques et algorithmes de filtrage.

Solution en PPC

- Advantages des variables “digraph” :
  - Simple
  - Notion de connexité immédiatement intégrée
  - Permet la définition de contraintes additionnelles
  - Bien plus efficace que le modèle classique de PPC (une variable par noeud, arcs definit par le domaine)

Résultats

- France Telecom considère que la PPC donne les meilleures résultats.
- L'approche PPC a été principalement optimisé pour la série A.

Plan

- Principes de la Programmation par Contraintes
- Problématique de la modélisation
- Algorithme de filtrage et contraintes globales
- Problèmes sur-contraints
- Applications
- **Un exemple détaillé : la contrainte de séquence**
- Perspectives
- Conclusion

Car sequencing

- Problème : calculer l'ordre selon lequel les voitures doivent être construites sur une chaîne de montage
- Différents types de voitures peuvent être construites
- **Une voiture = une voiture de base + options** (couleur, moteur, boîte de vitesse, …).
- **Une voiture = une configuration d'options**

Capacité d'une option

- Pour des raisons pratiques une option donnée ne peut pas être mise sur toute les voitures de la chaîne.
- **Capacité d'une option** : rapport p/q, pour n'importe quelle séquence de q voitures consécutives au plus p auront l'option

Car sequencing

opt	cap	configurations					
		0	1	2	3	4	5
0	1/2	X				X	X
1	2/3			X	X		X
2	1/3	X				X	
3	2/5	X	X		X		
4	1/5			X			
#voitures			1	1	2	2	2	2

Car sequencing

- opt cap configurations

		0	1	2	3	4	5
0	1/2	X				X	X
1	2/3			X	X		X
2	1/3	X				X	
3	2/5	X	X		X		
4	1/5			X			
#voitures		1	1	2	2	2	2

- Séquences 4,4 or 4,5 ou 0,4 ou 0,5 sont interdites

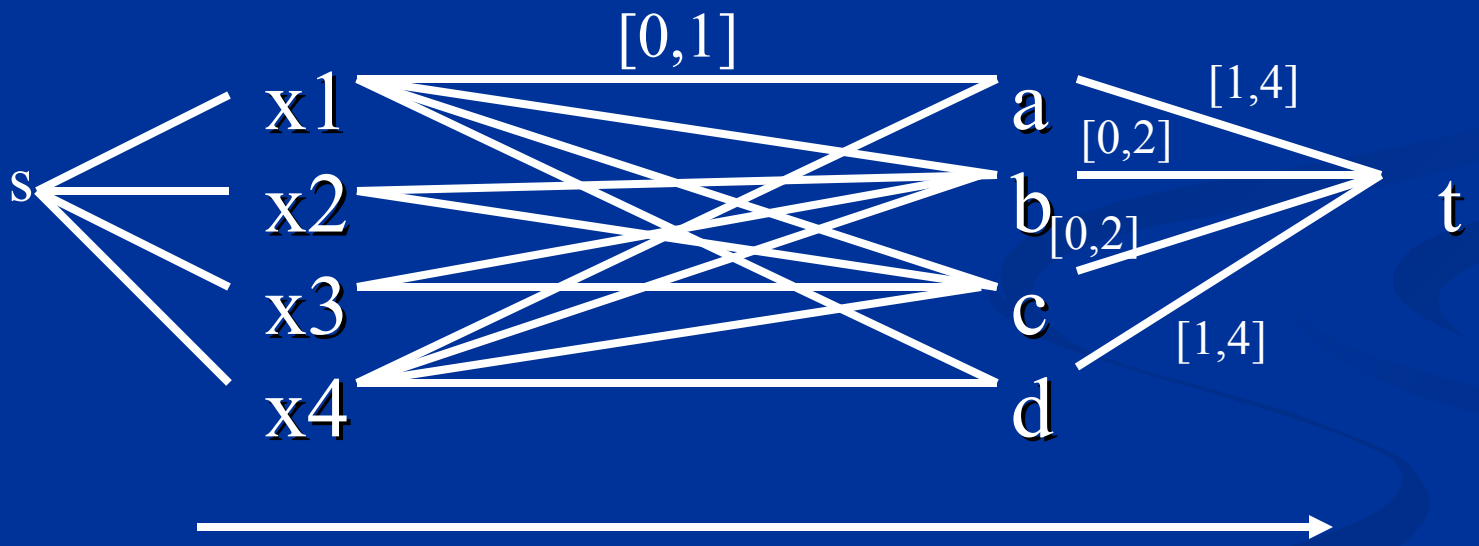
Car sequencing

opt	cap	configurations					
		0	1	2	3	4	5
0	1/2	X				X	X
1	2/3			X	X		X
2	1/3	X				X	
3	2/5	X	X		X		
4	1/5			X			
#voitures		1	1	2	2	2	2

- Séquences 2,2,1 ou 2,3,0 sont autorisées
- Séquences 2,2,3 ou 5,3,2 sont interdites

GCC : Algorithme de filtrage

- Peut être représenté par un flot :

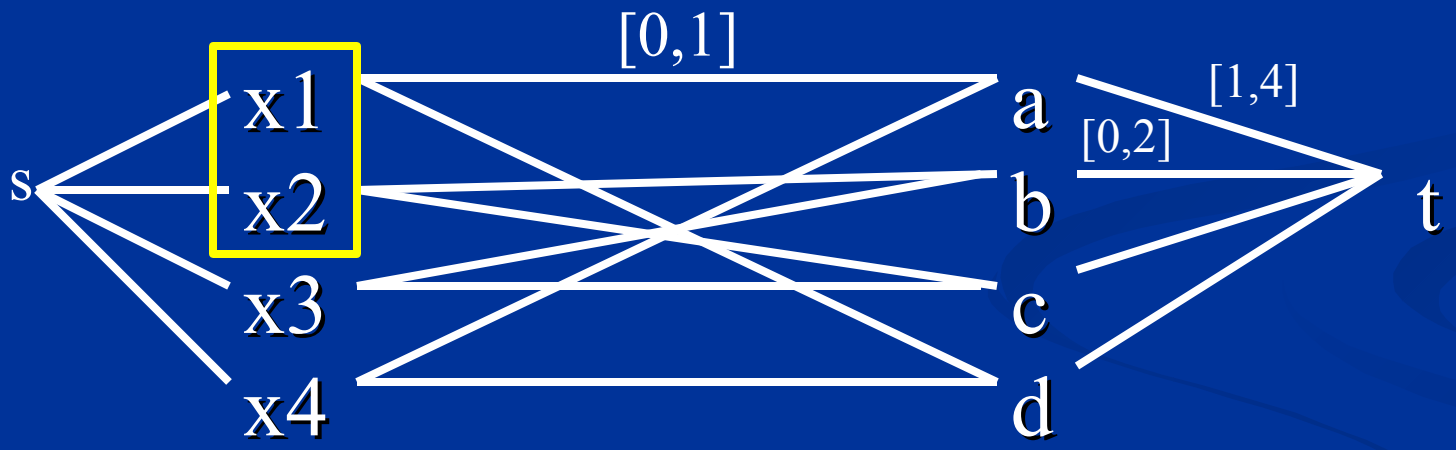


Valeur du flot: |X| =4

Contrainte de Séquence Globale

- GSC(X,V,min,max,q,{li},{ui})
- Une GCC (contrainte de cardinalité globale) pour les valeurs de V + une contrainte imposant que pour chaque séquence de q variables consécutives, au moins min et au plus max variables de S prennent leur valeur dans V.
- But : **Intégrer la séquence dans la GCC**

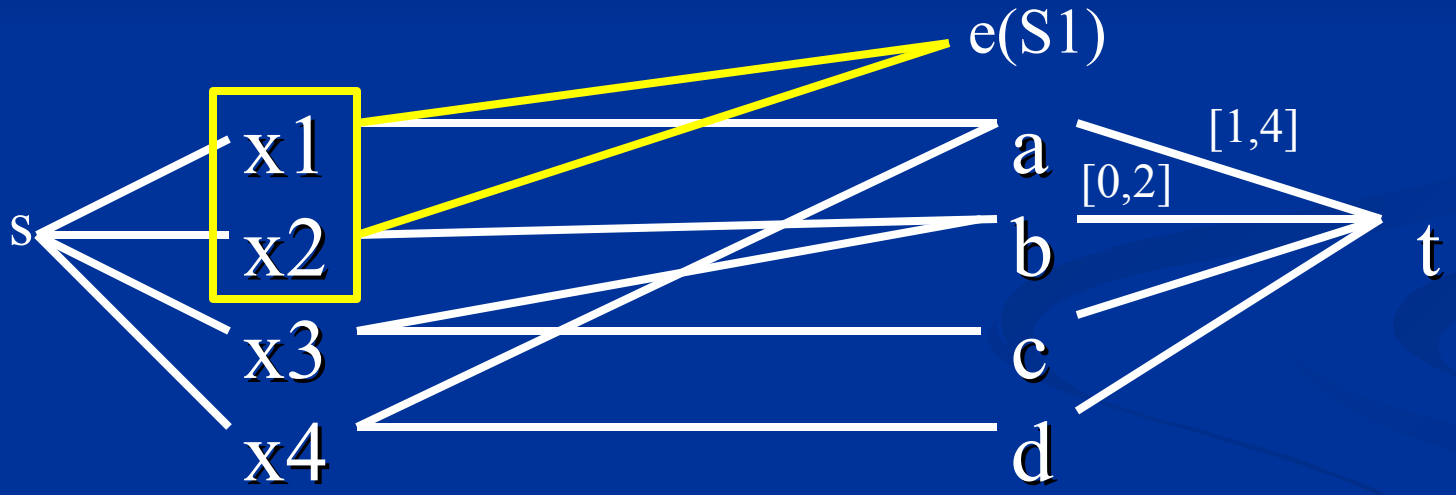
- GSC(X,V={a,b},min=0,max=1,q=2,…)



Les Contraintes sur les valeurs ∉ V ne sont plus considérées

Valeurs abstraites

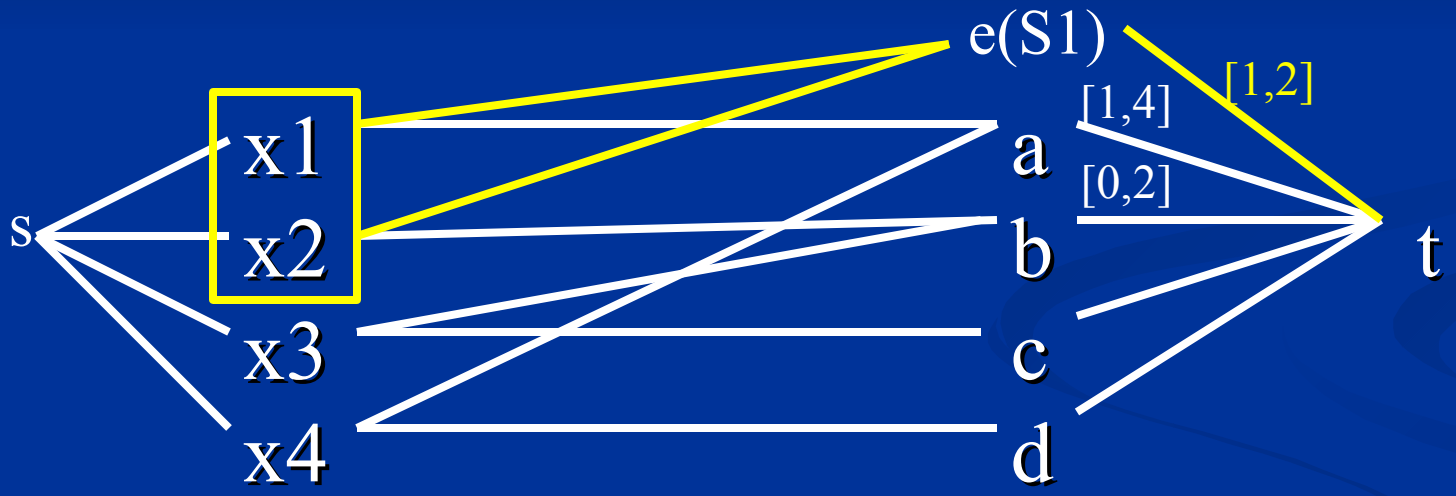
- GSC(X,V={a,b},min=0,max=1,q=2,…)



Les valeurs c et d ∉ V, elles sont remplacees par e(S1), pour la sequence

Valeur abstraite

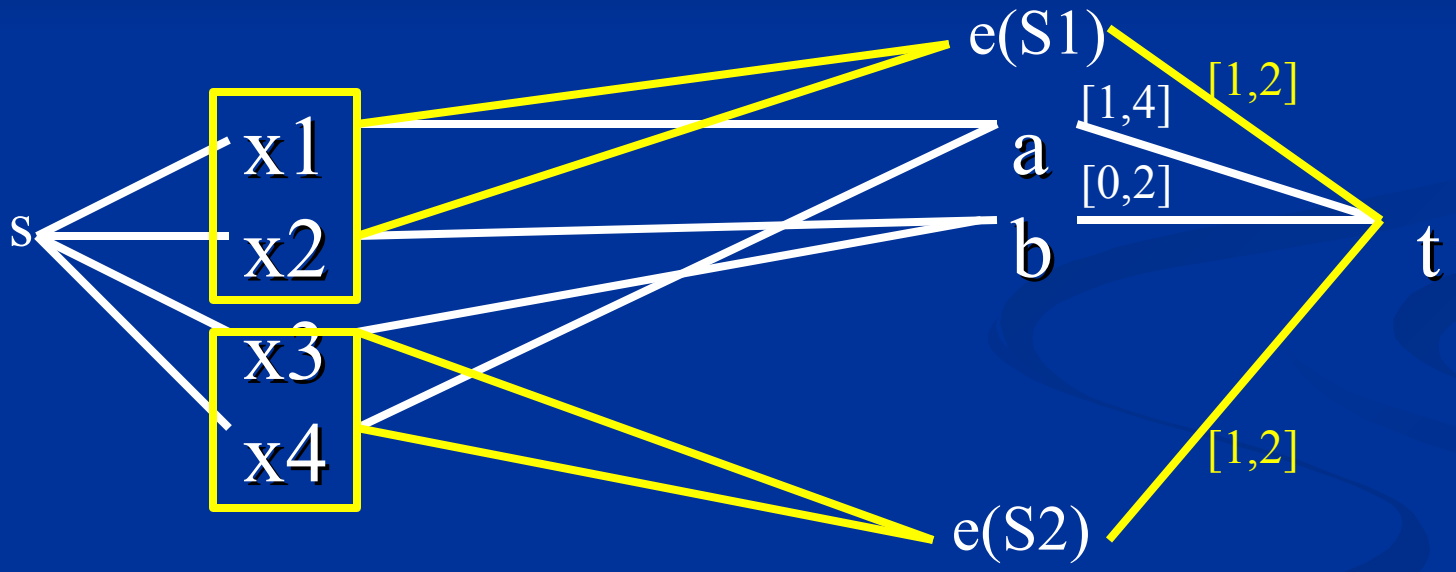
- GSC(X,V={a,b},min=0,max=1,q=2,…)



La valeur e(S1) doit etre prise au moins |S|-max=2-1=1 fois et au plus q -min = 2-0=2 fois

Eclatement de X en une partition de séquences

- GSC(X,V={a,b},min=0,max=1,q=2,…)



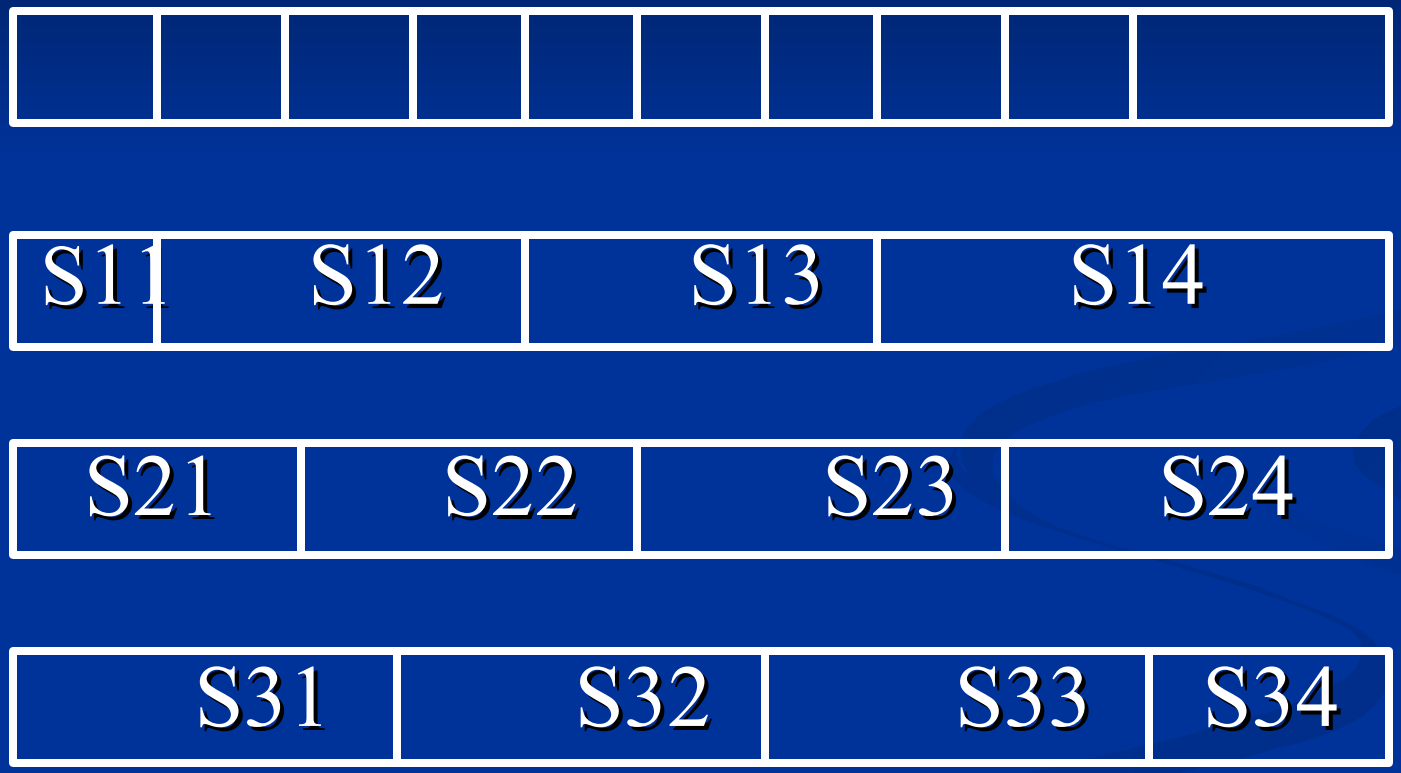
Les arcs jaunes repersentent les contraintes sur les sequences
Les arcs blancs representent les contraintes sur les valeurs de V

Eclatement de X en une partition de séquences

- Problème : un nombre exponentiel de séquences existe
- Solution : on a seulement besoin de représenter chaque séquence une.
  Proposition : |X| partitions simultanées.
- Pour notre exemple : P1={(x1,x2),(x3,x4)} et P2={(x1),(x2,x3),(x4)}

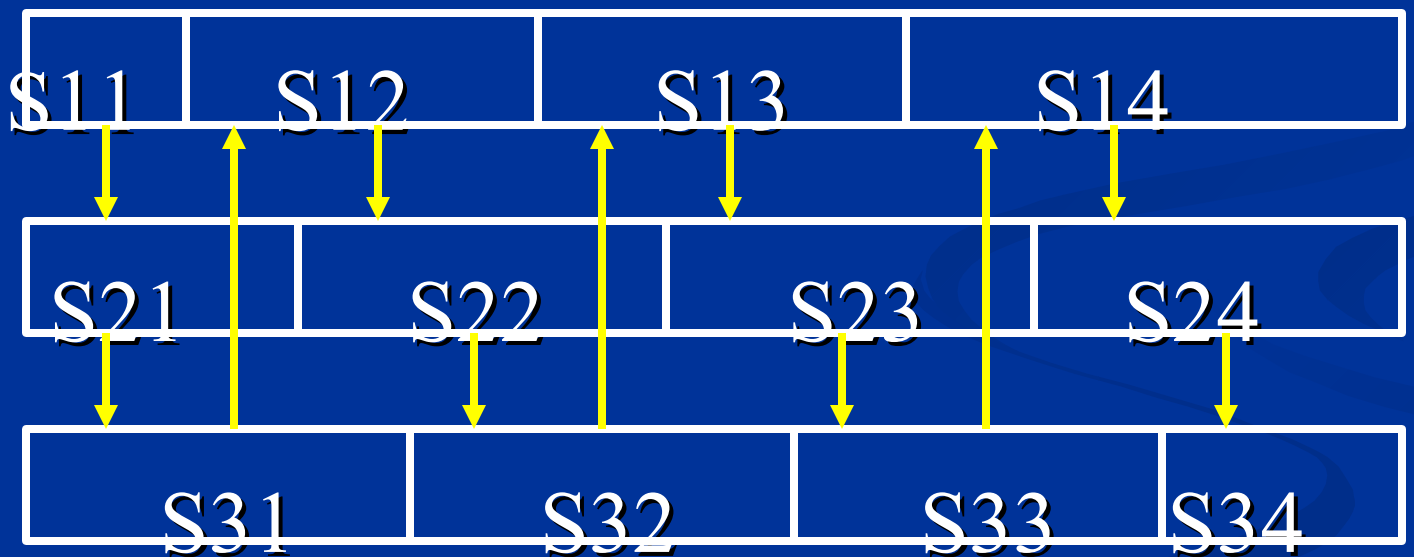
Partitions de séquences

- x1 x2 x3 x4 x5 x6 x7 x8 x9 x10



Partitions de sequences

- x1 x2 x3 x4 x5 x6 x7 x8 x9 x10



Si ⟶ Sj signifie que Sj est un successeur de Si

Contraintes entre les séquences

- $|\#e(S1) - \#e(S2)| \leq 1$ doit être implémentée minutieusement :
  S1: x1 x2 x3
  S2: x2 x3 x4

- $x1=e(S1)$ et $x4 \neq e(S2) \Leftrightarrow \#e(S1) = \#e(S2) +1$
- $(x1=e(S1)$ et $x4=e(S2))$ ou $(x1 \neq e(S1)$ et $x4 \neq e(s2)) \Leftrightarrow \#e(S1) = \#e(S2)$
- $x1 \neq e(S1)$ et $x4=e(S2) \Leftrightarrow \#e(S1) = \#e(S2) - 1$

Car sequencing

- Pour chaque option une contrainte globale de séquence est utilisée
- Cette méthode a permis de fermer certains problèmes ouverts
- Elle est toujours la seule à être capable de prouver que certains problèmes n'ont pas de solution.

Plan

- Principes de la Programmation par Contraintes
- Problématique de la modélisation
- Algorithme de filtrage et contraintes globales
- Problèmes sur-contraints
- Applications
- Un exemple détaillé : la contrainte de sequence
- **Perspectives**
- Conclusion

Perspectives

- TSP (voyageur de commerce)
- Combinaison de contraintes (cardinalité et table, alldiff et ordre …)
- Amélioration de la complexité algorithmes de filtrage (symmetric alldiff)
- Nouveaux algorithmes de filtrages (séquence)
- Contraintes NP-Complètes
- Problème de contiguïte (all min distance, stretch, pb d'ordonnancement)

Plan

- Principes de la Programmation par Contraintes
- Problématique de la modélisation
- Algorithme de filtrage et contraintes globales
- Problèmes sur-contraints
- Applications
- Un exemple détaillé : la contrainte de sequence
- Perspectives
- **Conclusion**

Conclusion

- J'ai introduit les algorithmes de filtrages pour les contraintes de tables (CAC, GAC-Schema) et pour les contraintes les plus utilisées en pratique (alldiff, cardinalité globale avec et sans coûts)
- J'ai défini des algorithmes de filtrages pour de nombreuses contraintes (séquence, allmin distance, symmetric alldiff, sum +alldiff, produit scalaire + alldiff (ou cardinalité), sum + inégalité binaire, partition, allnullintersect …)

Conclusion

- J'ai proposé un nouveau modèle pour les problèmes sur-contraints. Ce modèle permet de traiter les problèmes réels
- J'ai proposé des algorithmes de filtrages originaux utilisant un nouveau point de vue (les ensembles de conflits) pour les problèmes sur-contraints. Ces algorithmes permettent d'utiliser les algorithmes de filtrage des contraintes même quand les contraintes sont molles.
- J'ai introduit la notion de contrainte globale molle et deux définitions générales du coût de violations d'une contrainte

Conclusion

- J’ai obtenu de bons résultats pour des problèmes complexes : tournois sportifs, recherche de la plus grande clique dans un graphe, dimensionnement de réseau, car sequencing. Pour certains de ces problèmes, le modèle que j’ai proposé n’est toujours pas battu

Conclusion

- J'ai encadré un thésard (T. Petit) avec C. Bessiere, qui est maintenant en poste à l'école des Mines de Nantes
- J'encadre actuellement un autre thésard (D-O Fernandes-Pons) avec M. Minoux
- J'ai travaillé seul et aussi avec de nombreux chercheurs reconnus du domaine

Conclusion

- Je remercie la société ILOG de m'avoir permis de faire tout cela. Je remercie aussi tous les gens brillants qui y travaillent

Conclusion

- J’ai encore plein d’idées.