Programmation par Contraintes

Jean-Charles Régin

Univ. Nice-Sophia Antipolis, I3S-CNRS

Résolution de problèmes

- Soit le problème est « polynomial », soit pas
- La PPC (programmation par contraintes) s'intéresse aux problèmes NP-Complets, donc difficiles
  - Par exemple $\sum x_i = s$ avec x_i discrets ({0,3,6,7,9..})
  - On veut mettre des machines virtuelles dans des serveurs en respectant le ressources (RAM, CPU,…) en minimisant le nombre de serveurs utilisés

Résolution de problèmes difficiles

- **Greedy** (algorithme glouton) :
  - on applique une règle pour le prochain élément (je met la MV dans le serveur le plus adapté)
  - On ne remet JAMAIS en cause les choix
- **Iterative Improvement**
  - On calcule une solution et puis on essaie de faire des échanges qui l'améliore (on maximise la taille libre du serveur le moins utilisé)
- **Local search, tabu search…**
- **Algorithmes approximés**
  - On sait calculer une solution à 19/18 de l'optimal

Résolution de problèmes difficiles

- Problèmes des méthodes précédentes
  - Très ad-hoc (ne marche pas bien dès que le problème est modifié)
  - L'optimal n'est jamais garanti. Parfois on l'atteint mais
    - On ne sait pas que c'est l'optimal
    - On ne peut pas prouver que c'est optimal
- Solution : utilisation de méthode exacte
  - Enumération intelligente

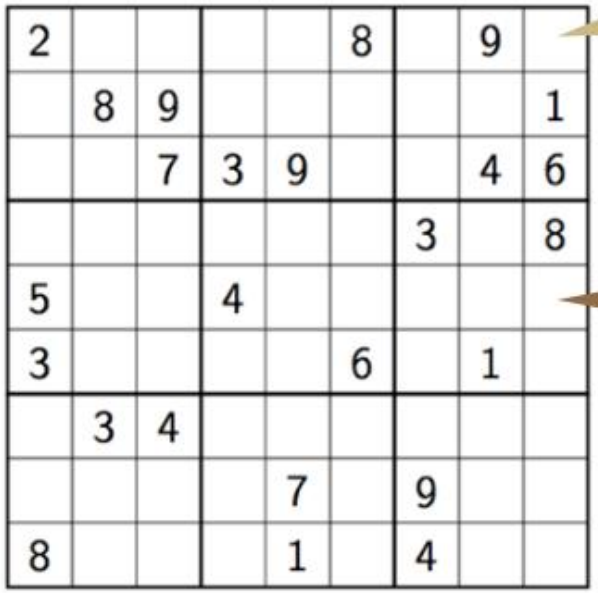
PPC ou CP

- La Programmation Par Contraintes (PPC) ou Constraint Programming (CP) a pour but de résoudre des problèmes réels
  - Résolution exacte même en flottants (on gère des intervalles)
  - Enumération exhaustive (mais intelligente) de l'espace de recherche
  - Les problèmes peuvent être difficiles (NP-Complet)
    - On ne peut pas tous les résoudre
    - On peut aider à la résolution

Sudoku : raisonnement simple

2					8		9	
	8	9						1
?		7	3	9			4	6
						3		8
5			4					
3					6		1	
	3	4						
				7		9		
8				1		4		

Sudoku et CP



2					8		9	
	8	9						1
		7	3	9			4	6
						3		8
5			4					
3					6		1	
	3	4						
				7		9		
8				1		4		

une case = une variable

valeurs possible : 1 .. 9

trois *contraintes*

- tous les chiffres de 1 à 9 pour chaque ligne
- tous les chiffres de 1 à 9 pour chaque colonne
- tous les chiffres de 1 à 9 pour chaque bloc

un but

trouver **la** solution

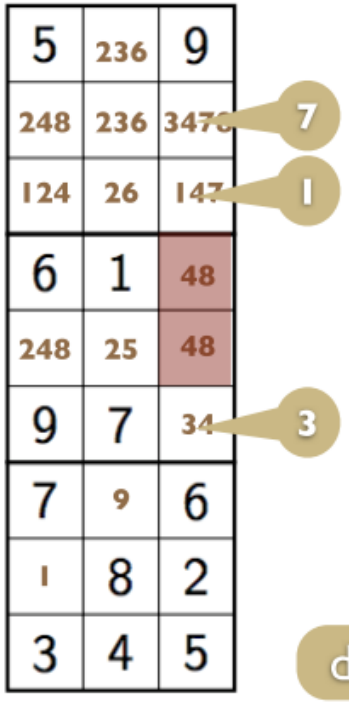
2					8		9	
	8	9						1
		7	3	9			4	6
						3		8
5			4					
3					6		1	
	3	4						
				7		9		
8				1		4		

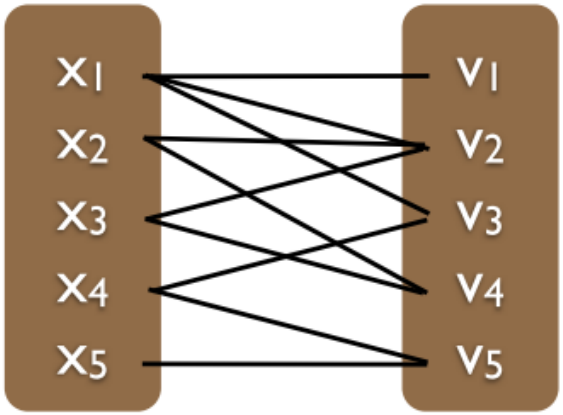
pourquoi devrait-on faire plus que décrire le problème ?

programmer avec des contraintes

```
int n = 9;
Problem pb = new Problem(); // a new instance
IntVar[][] rows = new IntVar[n][]; // the n rows
IntVar[][] cols = new IntVar[n][]; // the n columns
IntVar[][] blocks = new IntVar[n][];
for (int i = 0; i < n; i++) {
  pb.post(pb.allDifferent(cols[i]));
  pb.post(pb.allDifferent(rows[i]));
  pb.post(pb.allDifferent(blocks[i]));
}
pb.post( pb.eq(rows[4][6], 5));
pb.solve();
```

Sudoku : raisonnement plus complexe





non

quel est le couplage maximal ?

déduction

quels sont les arcs dans **tous** les couplages maximaux ?

réduction

quels sont les arcs dans **aucun** couplage maximal ?

Principes généraux

2					8		9	
	8	9						1
		7	3	9			4	6
				1 2 3 4 5 6 7 8 9		3		8
5			4					
3					6		1	
	3	4						
				7		9		
8				1		4		

1. raisonnement **local**
   satisfiabilité
   réduction de domaine
2. **propagation**
   interaction entre contraintes
3. **réduction** de domaine
   maintenir et partager info
4. itération jusq'à un
   point fixe

Constraint Programming

- Modélisation intuitive = proche de la réalité
  - Pas besoin de transformer |x-y|=z en contraintes linéaires, ni en formule SAT
  - Variables discrètes naturellement, variables ensemblistes
- Résolution d'un problème P contenant les sous problèmes Q et R : basée sur la résolution de Q et de R (réutilisation de l'existant)

Résolution de Problèmes Réels

- Entreprises qui développent leur solveur de CP :
  - Google
  - Microsoft
  - IBM
  - SaS
- Entreprises qui utilisent la CP pour résoudre leurs problèmes
  - SAP
  - Intel
  - Oracle

Résolution de Problèmes réels

- Quelques exemples :
  - Car sequencing (ligne virtuelle chez Nissan, amélioration du paint shop,…)
  - Dimensionnement de réseau de télécom
  - Remplissage d'avions cargos
  - Construction de barrages
  - Minimisation du nombre d'armoires électriques dans les centrales nucléaires
  - Remplissage de camions et tournée de véhicules
  - Sports Scheduling
  - Emploi du temps…

Résolution de benchmark

- On s'intéresse aussi aux benchmarks classiques
  - TSP
  - Clique Max

Avantages de la CP

- Apporte une grande culture informatique
  - Utilisation et besoin réel de toute la puissance des ordinateurs
  - Vaste choix de méthodes (local search, heuristic,…)
  - La CP est basée sur l'utilisation d'algorithmes efficaces (flots, couplages, automates,…)

Avantages de la CP

- Possibilité de laisser libre cours à son imagination
  - Attention: cela ne veut pas dire faire n'importe quoi
  - Vous devez apporter des résultats et vous comparez aux autres
  - Vous pourrez montrer que ce vous faites est mieux que les autres.
- Plaisir de fermer pour la première fois un problème ouvert
- Utilisation plus naturelle et plus puissante que les autres méthodes de résolutions (MIP, SAT, Local Search)

CP

- Convient bien aux gens ayant un profil un peu « joueur »: ancien champion d'Othello, joueurs d'échecs…
- Vous devez être imaginatif
- Vous devez savoir programmer

CP

- Vient de plusieurs domaines
  - IA (Intelligence Artificielle)
  - RO (Recherche Opérationnelle)
  - Langage (PROLOG)

Histoire

- General Problem Solvers in 70's
  - ALICE [J-F Lauriere, AIJ 78], phD in Paris VI, 76
- Prolog: CHIP, ECRC Munich 86 (Alice in Prolog)
  - Colmerauer, Gallaire, Van Hentenryck
- Constraint Satisfaction Problems
  - Waltz 72, Mackworth 74, Freuder 76
- Industry: Bull (Charme), Cosytech, ILOG (Solver) 92

Solver

- Un Solver est un moteur de résolution de problèmes
  - On définit le problème
  - On explique comment le résoudre
  - Le solver fait la résolution (on code peu)
- De nombreux solvers basés sur la CP sont disponibles:
  - Choco
  - Comet,
  - Gecode,
  - Minion,
  - Or-tools …

Programmation par Contraintes

- En CP un problème est défini par:
  - des variables ayant des valeurs possibles (domaine). D(x) est le domaine de la variable x
  - contraintes
- Les domaines peuvent être discrets, continus, avec des valeurs symboliques ou numériques
- Les contraintes expriment des propriétés qui doivent être satisfaites

Problème = conjonction de sous-problèmes

- En CP un problème peut être vu comme une conjonction de sous-problèmes que l'on est capable de résoudre (donc faciles)
- Un sous-problème peut-être trivial, comme $x < y$, ou complexe comme la 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 encapsulé
- Combinaison logique de contraintes en utilisant les opérateurs OR, AND, NOT, XOR. Parfois appelées méta-contraintes

Filtrage

- Nous sommes capable de résoudre un sous-problème : une méthode est disponible
- CP utilise cette méthode pour supprimer des valeurs des domaines qui n'appartiennt pas à une solution de ce sous-problème. Ce mécanisme est appelé **fitrage ou réduction de domaine**
- Ex: x < y and D(x)=[10,20], D(y)=[5,15]
  => D(x)=[10,14], D(y)=[11,15]

Filtrage

- Un algorithme de filtrage est associé avec chaque contrainte (sous-problème).
- Peut être simple ($x < y$) ou complexe (alldiff)
- Aspect théorique : **arc consistance** (cohérence d'arc): supprime toutes les valeurs qui n'appartiennent pas à une solution du problème sous-jacent.

Arc consistance

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

Propagation

- La réduction de domaine due à une contrainte peut conduire à de nouvelles réduction de domaines pour les autres variables
- Quand un domaine est modifié alors toutes les contraintes impliquant la variable doivent être réétudié et ainsi de suite …

Propagation

- D(x)=D(y)={1,3}, D(z)=D(t)=D(u){0,1,2,3,4}
- Alldiff({x,y,z,t}), z < t, u=z+t, y=u+1
- z < t $\Rightarrow$ z $\neq$ 4 and t $\neq$ 0
- Alldiff(x,y,z,t) $\Rightarrow$ z $\neq$ {1,3} and t $\neq$ {1,3}
- u=z+t $\Rightarrow$ {0,2} + {2,4} $\geq$2 so u $\geq$2
- y=u+1 $\Rightarrow$ u=y-1={1,3}-1 $\leq$ 2 so u=2 and y=3
- Alldiff(x,y,z,t) $\Rightarrow$ x=1
- u=z+t $\Rightarrow$ 2=z+t $\Rightarrow$ z=0; t=2

Pourquoi Propager ?

- Un problème = conjonction de sous-problèmes faciles.
- Sous-problème : point de vue local. La propagation essaie d'obtenir un point de vue global à partir de points de vue globaux indépendants
- La conjonction est plus forte que l'union de résolutions indépendantes
- **Pour aider la propagation à avoir un point de vue global : on introduit des contraintes globales (i.e non locales) !**

Résultats pour la contrainte Alldiff

- Colorier le graph ayant 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}

	Local		Global	
Clique size	#fails	time	#fails	Time
27	1	0.17	0	1.21
31	65	0.37	4	2.26
51	24512	66.48	501	25.95
61	???	???	5	58.22

Procédure de recherche

- On utilise un algorithme de retour-arrière (backtrack) avec des stratégies :
  on essaie d'affecter successivement des valeurs aux variables. Si un échec se produit alors on remet en cause le dernier choix (on backtrack) et on essaie une autre valeur pour la variable.
- Stratégie : définit quelle variable et quelle valeur seront choisies pour la prochaine affectation.
- Après chaque affectation les algorithmes de filtrage et la propagation sont déclenchés

PPC

- 3 notions:
  - réseau de contraintes : variables, domaines
  contraintes + filtrage (réduction de domaine)
  - propagation
  - procédure de recherche (affectation + backtrack)

PPC

- On va tout étudier dans le détail
  - Les variables, avec leur domaine
  - Les contraintes
  - L'algorithme de recherche de solution