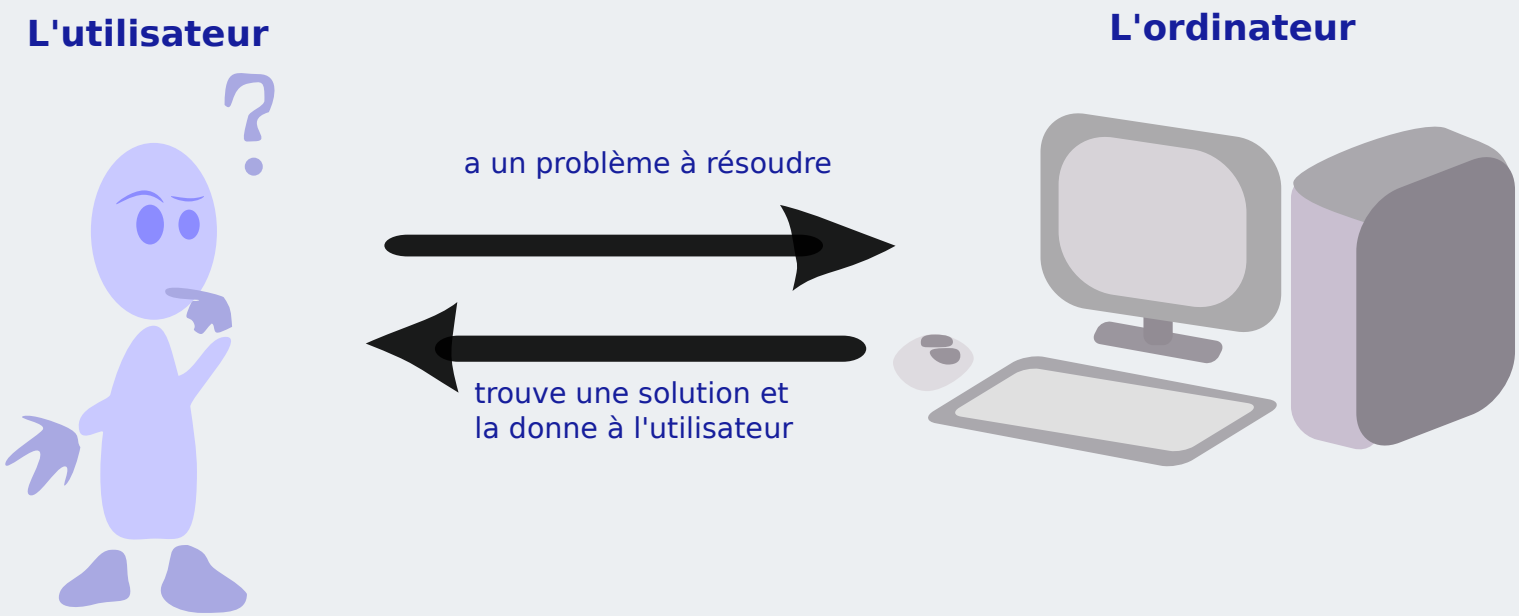
Apprentissage de problèmes de contraintes

Matthieu Lopez

sous la direction de **Arnaud LALLOUET**
réalisée au LIFO, Université d'Orléans,

8 décembre 2011

Intelligence Artificielle



Programmation par contraintes (CP) : exprimer/décrire un problème en spécifiant ses contraintes

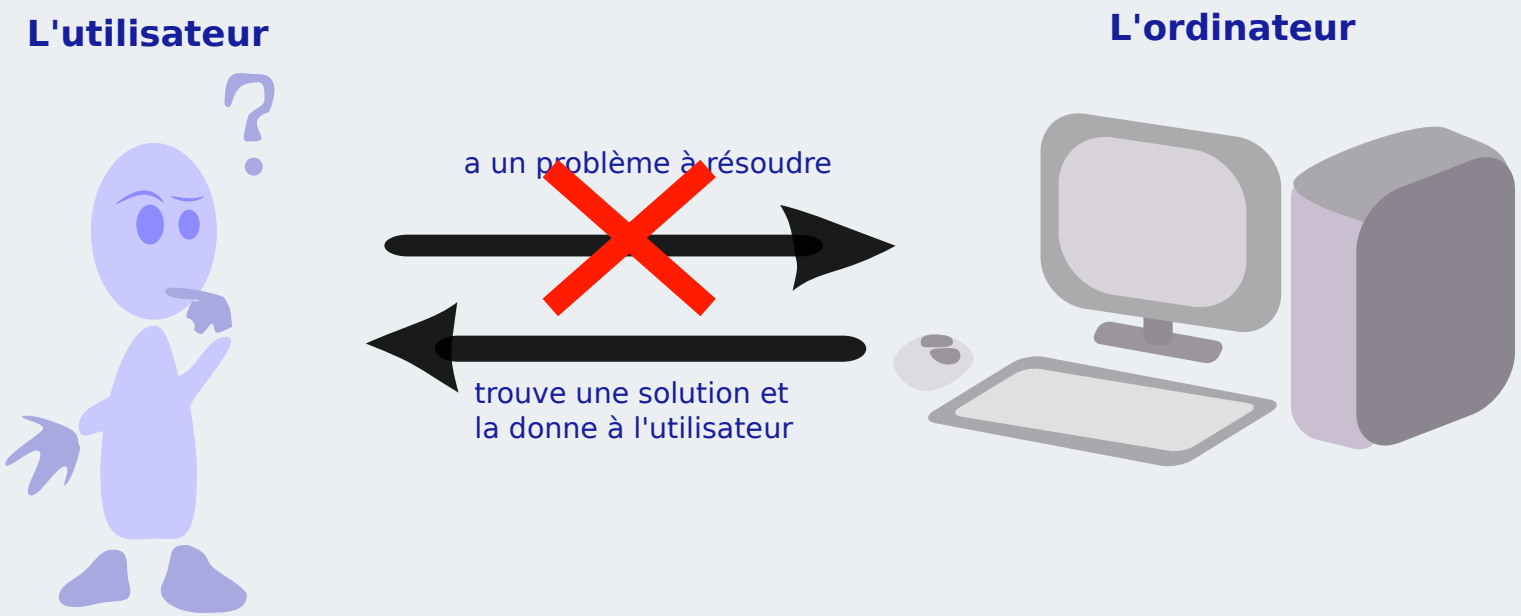
Exemple de l'emploi du temps scolaire.

- des cours, des professeurs, des salles et des créneaux horaires
- pas deux cours dans la même salle au même moment
- impossibilité pour un prof d'enseigner dans deux cours différents au même moment

Problème de satisfaction de contraintes (CSP)

- X : Ensemble de variables (ex. la salle pour le cours de math de M. Dupont)
- D : Ensemble de domaines (ex. {C205,C34,C109})
- C : Ensemble de contraintes

Programmation par contraintes



Limitations [Puget04] : modélisation trop complexe

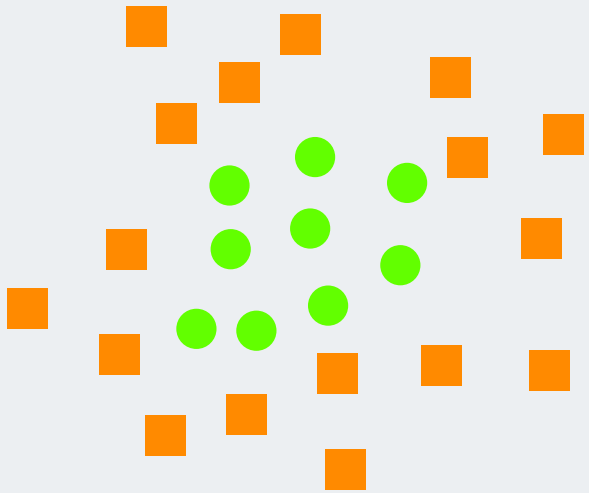
Comment rendre la CP plus accessible ?

Programmation par contraintes

Réponse de la communauté :

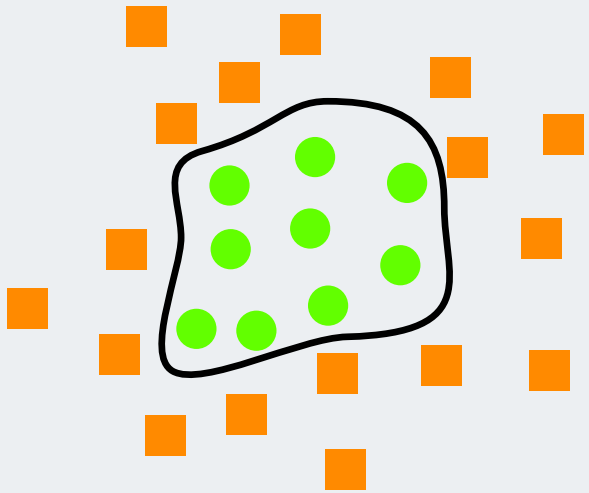
- langages de modélisation haut-niveau
- aide à la modélisation
- reformulation du modèle

Classification supervisée



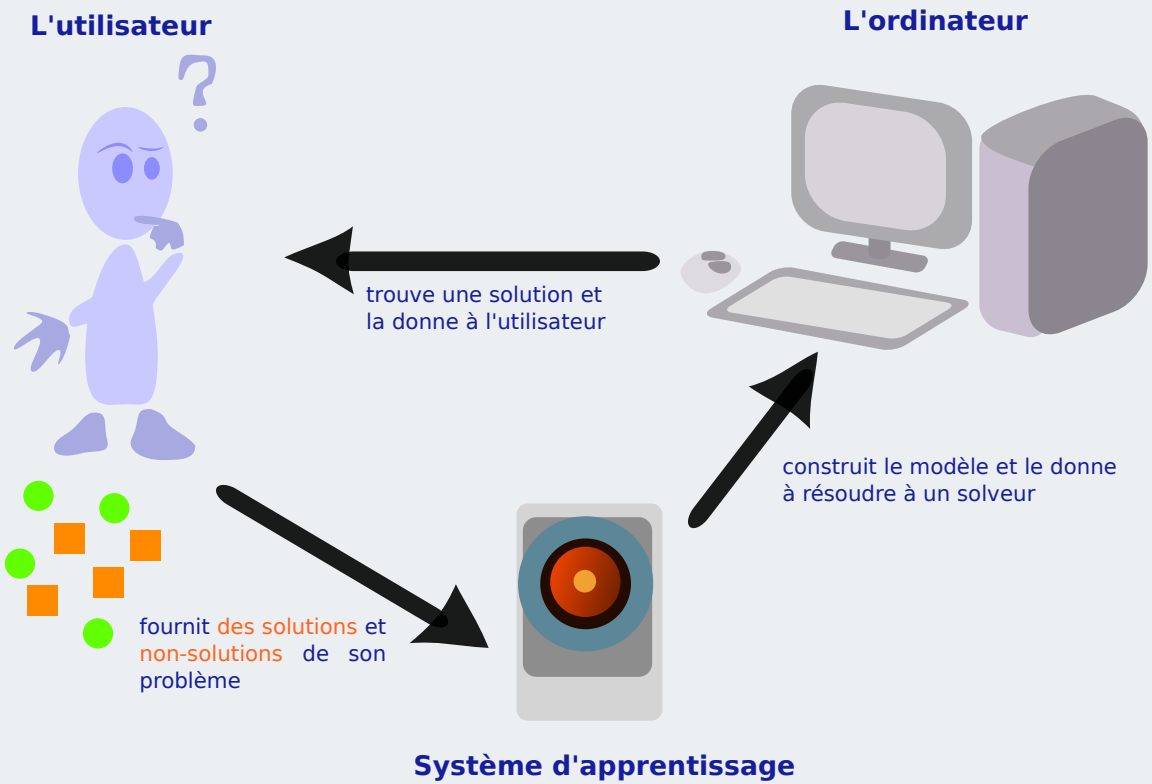
But : généraliser des données pour extraire un concept

Classification supervisée



But : généraliser des données pour extraire un concept

CONACQ [Bessière et al 2005 (et les suivants)]



CONACQ [Bessière et al 2005 (et les suivants)]

Un certain nombre de critiques :

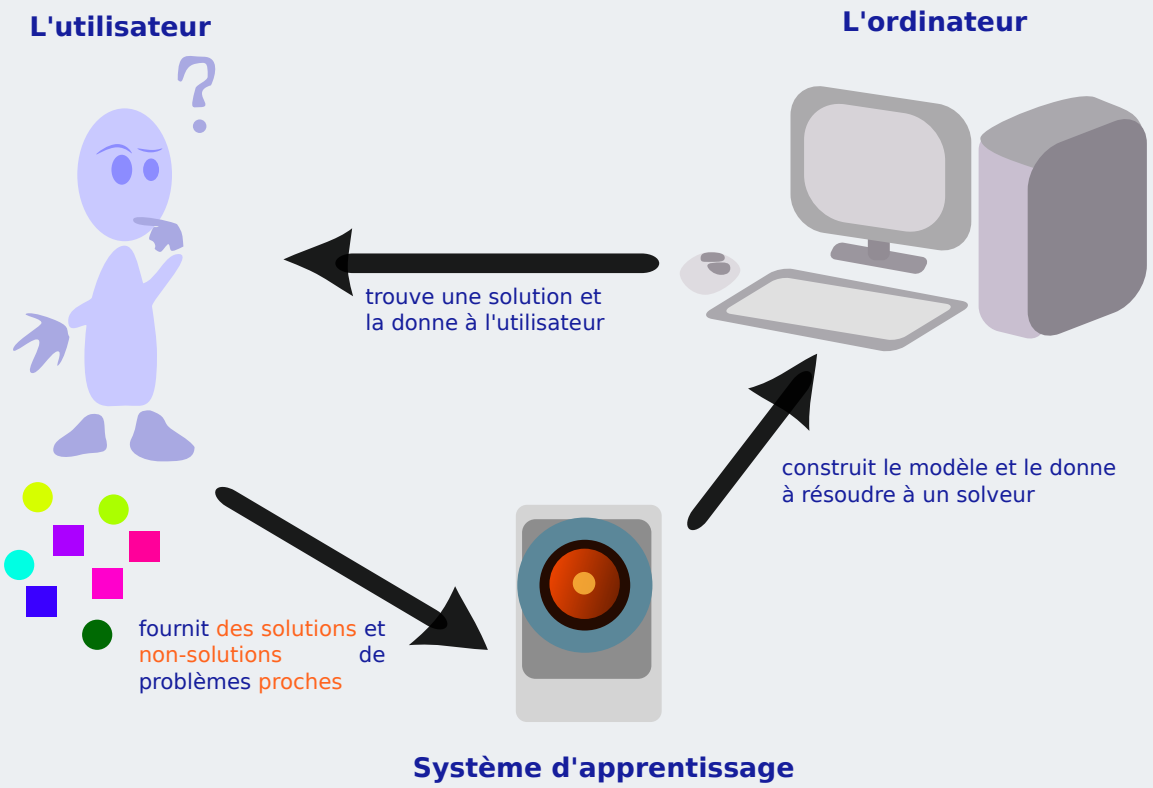
- fournir des solutions, quand on cherche une solution ?
- pourquoi apprendre un modèle si trouver des solutions est une tâche raisonnable ?

Objectif de cette thèse

Dépasser ces limitations !

$\Rightarrow$ 2 axes de recherche

Utiliser des solutions de problèmes proches



Problèmes proches ?

Emplois du temps des années passées $\Rightarrow$ exemples proches :

- le problème de fond reste le même
- mais de nouvelles données : nouveau prof, fermeture d'un bâtiment, réaménagement du temps scolaire

Les besoins ?

Trouver une solution

Apprendre le modèle n'est qu'une étape

$\Rightarrow$ pas obligatoire de l'achever

Axe de recherche : résolution interactive d'un CSP

1. Apprentissage de modèles abstraits
2. Générer-et-Tester et opérateur bi-directionnel
3. Recherche Interactive
4. Conclusion

Summary

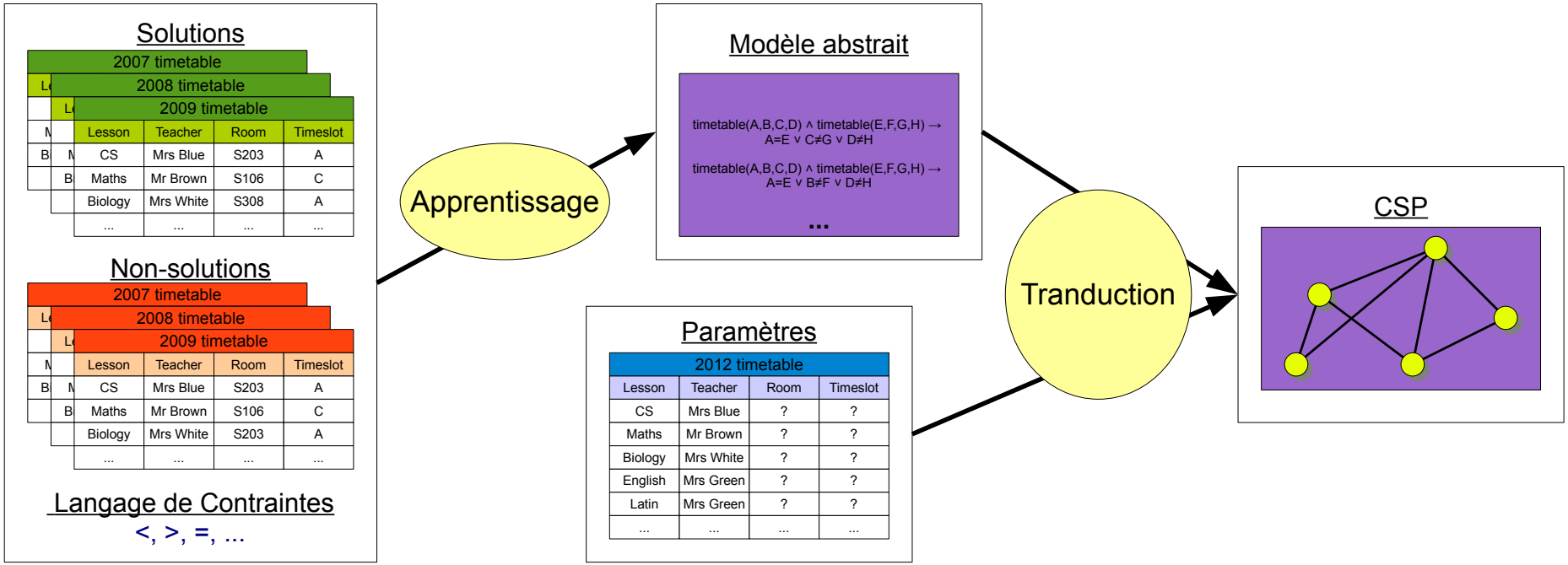
1 Apprentissage de modèles abstraits

2 Générer-et-Tester et opérateur bi-directionnel

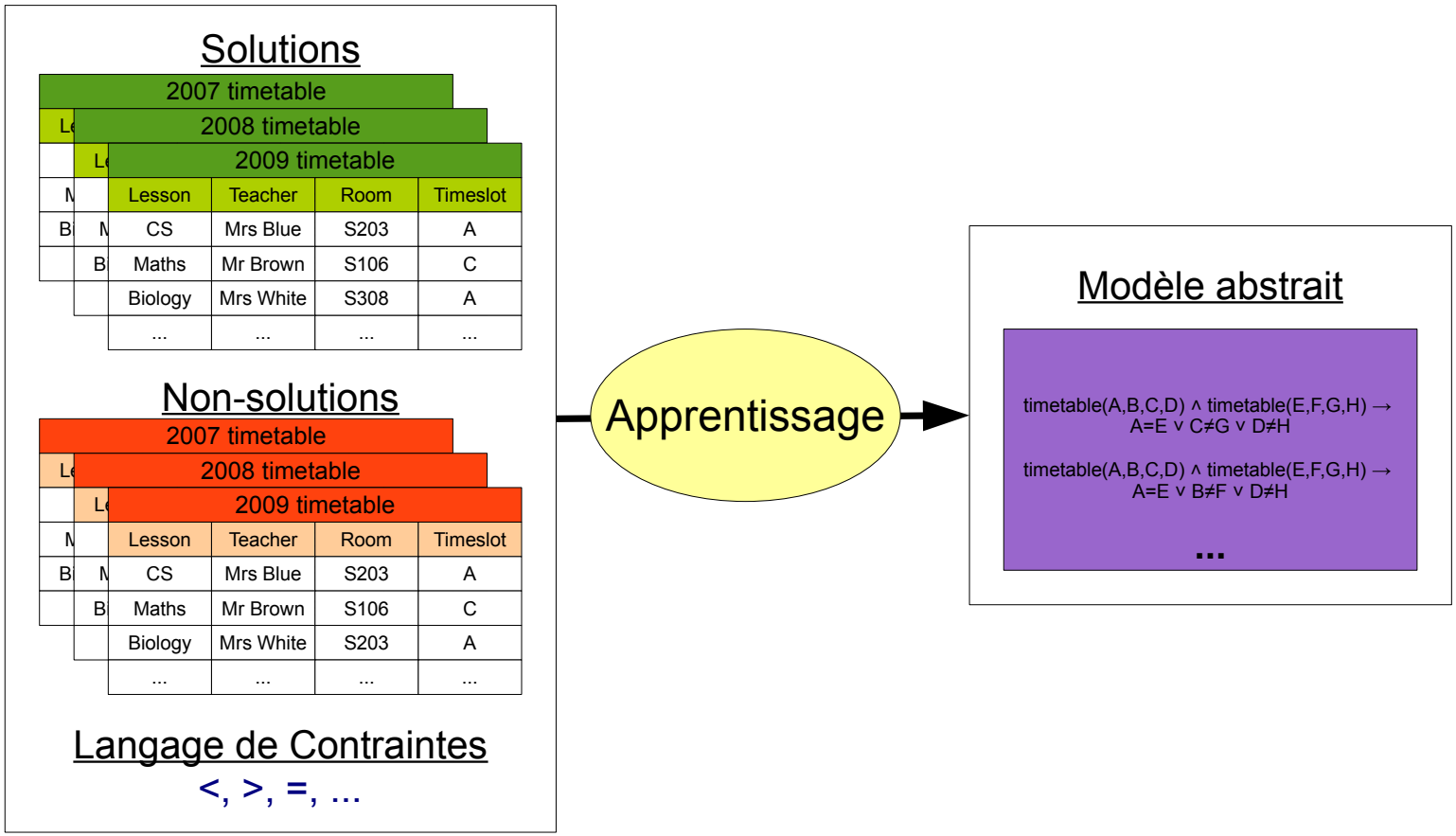
3 Recherche Interactive

4 Conclusion

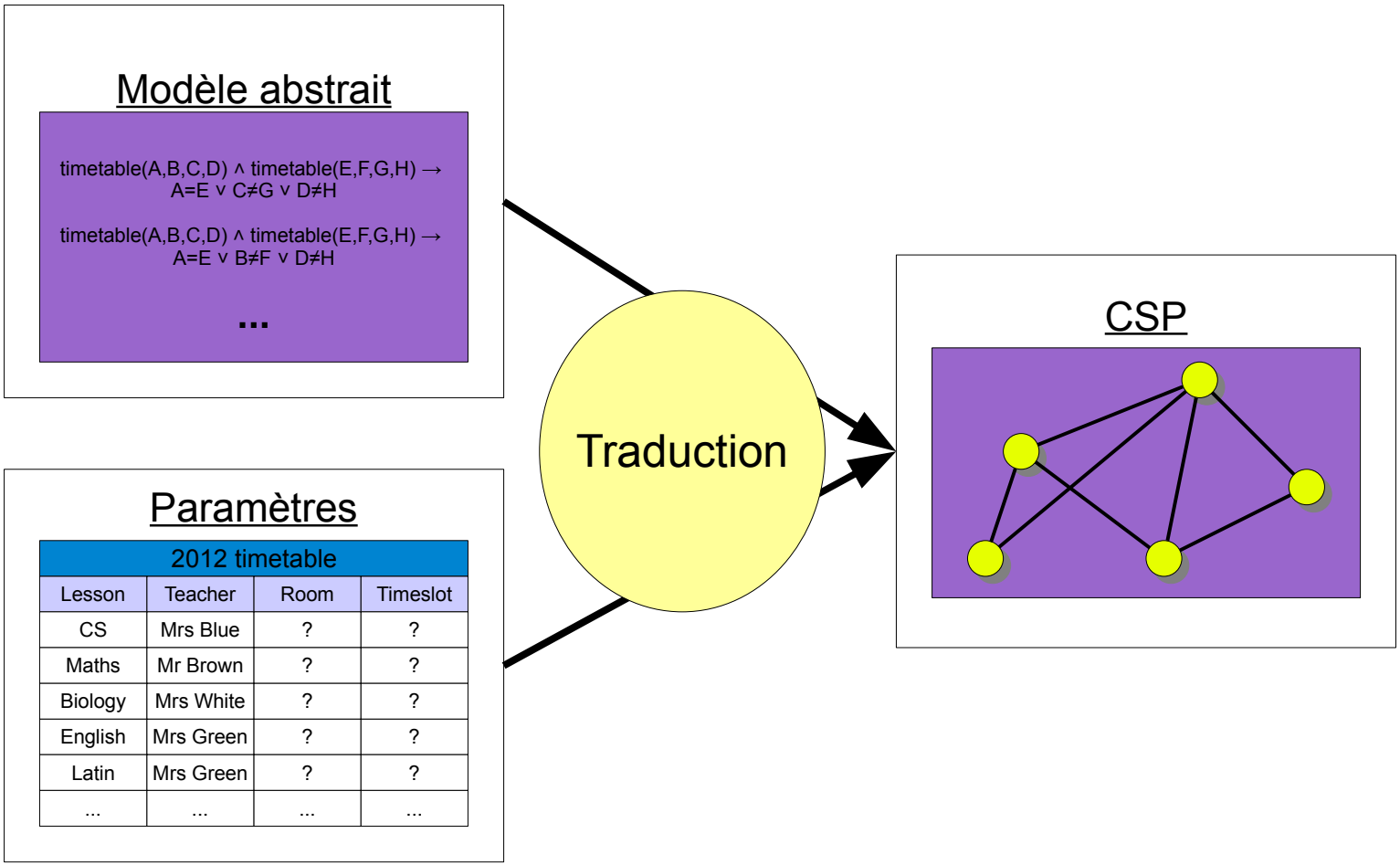
Le processus



Le processus



Le processus



2012 timetable			
Lesson	Teacher	Room	Timeslot
CS	Mrs Blue	?	?
Maths	Mr Brown	?	?
Biology	Mrs White	?	?
English	Mrs Green	?	?
Latin	Mrs Green	?	?
...	...	...	...

Des langages pour modéliser les CSP (OPL, Essence, Zinc)

$\Rightarrow$Pensés pour des utilisations humaines...

De quoi avons-nous besoin ?

- de décrire les contraintes du problème
- de ne pas considérer directement les variables du CSP
- d'être « apprenable »

$\Rightarrow$ Un sous-ensemble de la logique du premier ordre

- Atome : expression $p(t_1, \ldots, t_k)$,
  - p : prédicat
  - $t_1, \ldots, t_k$: termes (constantes ou variables)
- Littéral : atome ou négation d'un atome
- Littéral clos : littéral avec uniquement des constantes
- Connecteurs logiques : $\vee, \wedge, \rightarrow$
- Clause : disjonction de littéraux

Notre langage

Ensemble de règles de la forme :

$\forall$ *variables* : *body* $\longrightarrow$ *head*

- *body* : décrit les variables (du CSP) concernées par les contraintes
- *head* : décrit les contraintes qui seront produites (étape de réécriture)

Emploi du temps :

$$\forall L1, L2 \in Lesson,\ \forall T1, T2 \in Teacher,$$
$$\forall R1, R2 \in Room,\ \forall S1, S2 \in Timeslot\ :$$
$$timetable(L1, T1, R1, S1) \wedge timetable(L2, T2, R2, S2)$$
$$\longrightarrow T1 \neq T2 \vee L1 = L2 \vee S1 \neq S2$$

Les entrées de cette étape :

Le modèle abstrait

timetable(A,B,C,D) ∧ timetable(E,F,G,H) →
A=E ∨ C≠G ∨ D≠H

timetable(A,B,C,D) ∧ timetable(E,F,G,H) →
A=E ∨ B≠F ∨ D≠H

...

Les données incomplètes

2012 timetable			
Lesson	Teacher	Room	Timeslot
CS	Mrs Blue	?	?
Maths	Mr Brown	?	?
Biology	Mrs White	?	?
English	Mrs Green	?	?
Latin	Mrs Green	?	?
...	...	...	...

Les domaines

Room= {C205, C34, C109…}
Timeslot = {A, B, C...}

Les données incomplètes

2012 timetable			
Lesson	Teacher	Room	Timeslot
CS	Mrs Blue	?	?
Maths	Mr Brown	?	?
Biology	Mrs White	?	?
English	Mrs Green	?	?
Latin	Mrs Green	?	?
...	...	...	...

Les domaines

Room= {C205, C34, C109…}
Timeslot = {A, B, C...}

Les données incomplètes

2012 timetable			
Lesson	Teacher	Room	Timeslot
CS	Mrs Blue	X1 in Room	X6 in Timeslot
Maths	Mr Brown	X2 in Room	X7 in Timeslot
Biology	Mrs White	X3 in Room	X8 in Timeslot
English	Mrs Green	X4 in Room	X9 in Timeslot
Latin	Mrs Green	X5 in Room	X10 in Timeslot
...	...	...	...

Les domaines

Room= {C205, C34, C109…}
Timeslot = {A, B, C...}

Pour chaque règle

ex. :

$timetable(L_1, T_1, R_1, S_1) \wedge timetable(L_2, T_2, R_2, S_2) \rightarrow T_1 \neq T_2 \vee L_1 = L_2 \vee S_1 \neq S_2$

Recherche des substitutions du corps

$timetable(L_1, T_1, R_1, S_1) \wedge timetable(L_2, T_2, R_2, S_2)$

$\{L_1/Latin,\ T_1/Mrs\ Green,\ R_1/X5,\ S_1/X10,\ L_2/English,\ T_2/Mrs\ Green,\ R_2/X4,$
$S_2/X9\}$

Substitution de la tête puis ajout de la contrainte

$T_1 \neq T_2 \vee L_1 = L_2 \vee S_1 \neq S_2$

devient

$Mrs\ Green \neq Mrs\ Green \vee Latin = English \vee X4 \neq X9$

La contrainte peut se simplifier dans certains cas mais pas en général.

Cas spécifique : variables auxiliaires

Considérons :

- $val(X) \wedge val(Y) \wedge sum(X, Y, Z)$ dans le corps de la règle
- la substitution $\{X/2, Y/X1, Z/?\}$

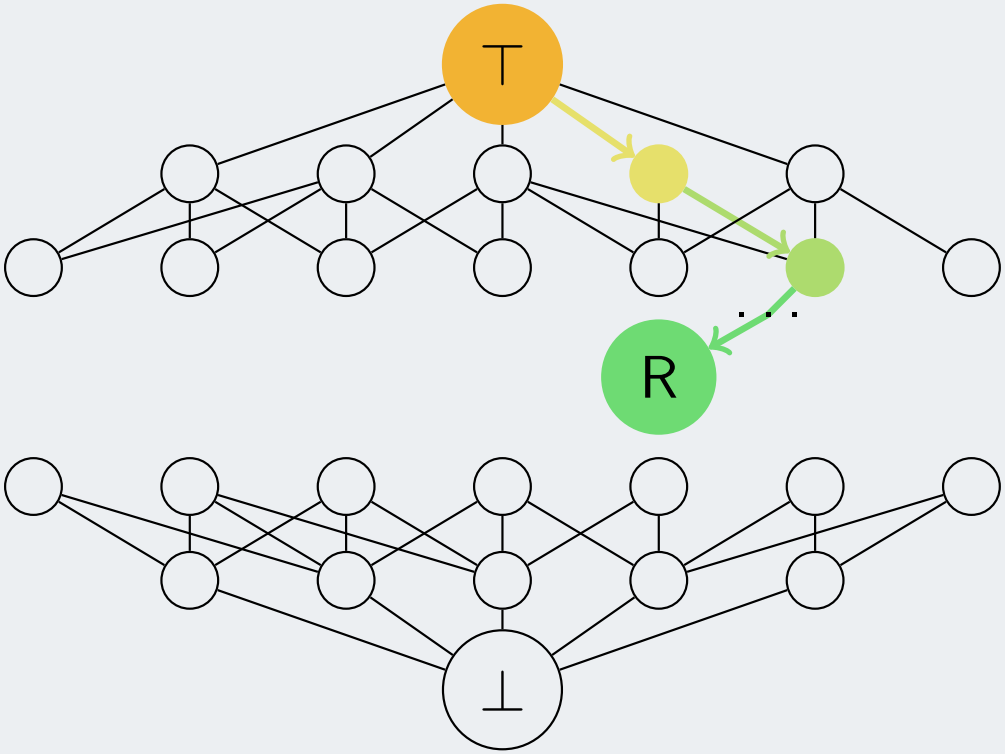
Comment remplacer Z ?

- création d'une nouvelle variable de CSP $X2$ *in* $[-inf, +inf]$
- ajout de la contrainte correspondant à $sum(2, X1, X2)$

Notre langage $\subset$ logique du 1er ordre

$\Rightarrow$ Programmation Logique Inductive (ILP)

Top-Down



Limites :

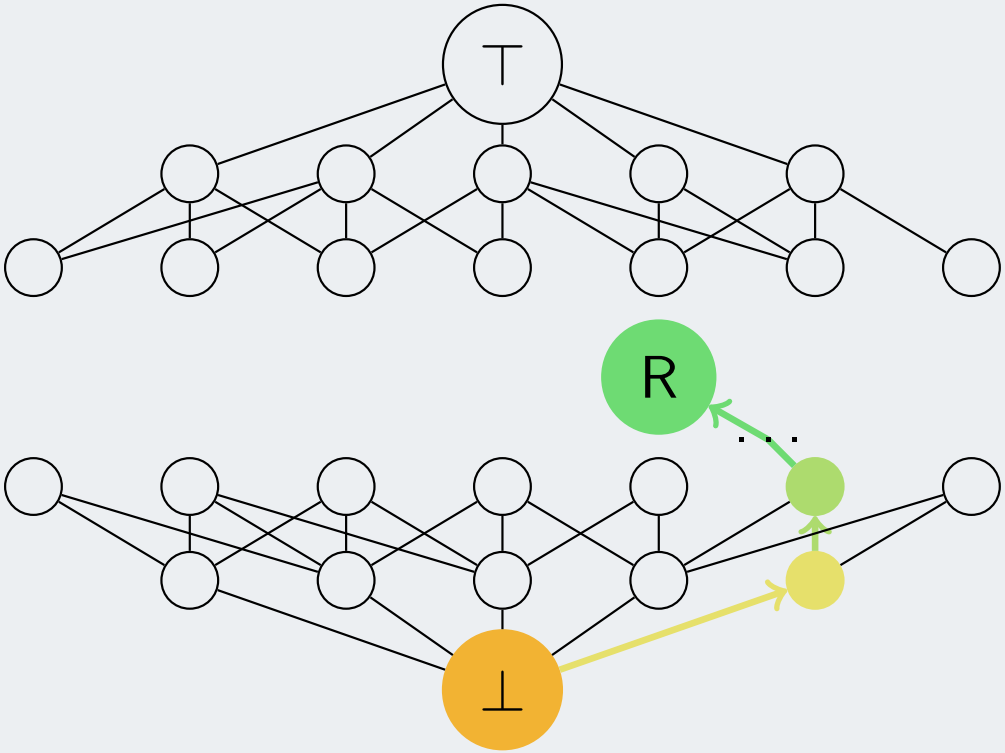
- Recherche aveugle
- Phénomène de plateau

Systèmes testés : Foil, Progol, Aleph, ICL, Beth, Propal

Notre langage $\subset$ logique du 1er ordre

$\Rightarrow$ Programmation Logique Inductive (ILP)

Bottom-Up



Limites :

- Test de subsomption trop coûteux

Systèmes testés : Progolem, Aleph

Pour résumer

- Changement des entrées : solutions/non-solutions de problèmes proches vs solutions/non-solutions du problème réel
- Passage par un modèle plus abstrait que le réseau
- Réécriture du modèle avec les données du problème réel
- Difficultés : problèmes pathologiques en ILP

Summary

1 Apprentissage de modèles abstraits

2 Générer-et-Tester et opérateur bi-directionnel

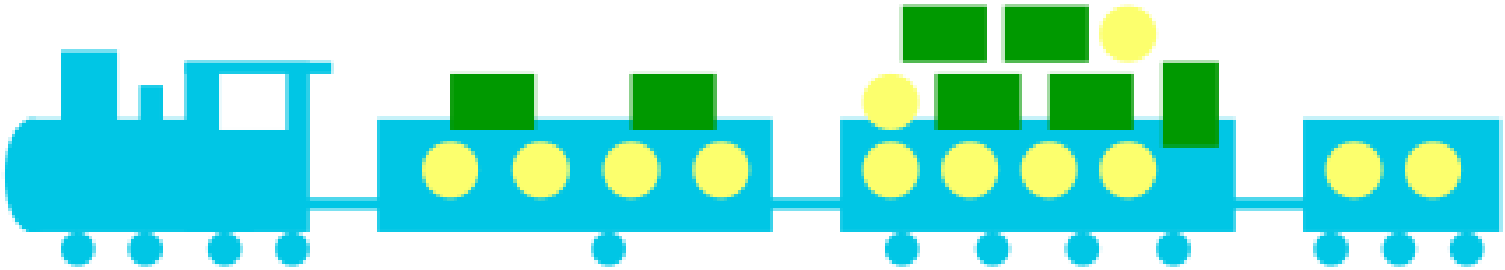
3 Recherche Interactive

4 Conclusion

Représentation des exemples

t^+ :

t^- :



t^+ :

Par implication (entailment)

$goodtrain(t^+)$

Dans la base de connaissances :

$has\_car(t^+, c_1), has\_car(t^+, c_2), has\_car(t^+, c_3), wheels(c_1, 2),$
$wheels(c_2, 3), wheels(c_3, 5), long(c_1), long(c_2), long(c_3),$
$load(c_1, circle, 3), load(c_2, circle, 6), load(c_3, triangle, 10)$

$sat(goodtrain(t^+), 3)$

Layer	Literals
0	$goodtrain(t^+)$

$sat(goodtrain(t^+), 3)$

Layer	Literals
0	$goodtrain(t^+)$
1	$has\_car(t^+, c_1), has\_car(t^+, c_2), has\_car(t^+, c_3)$

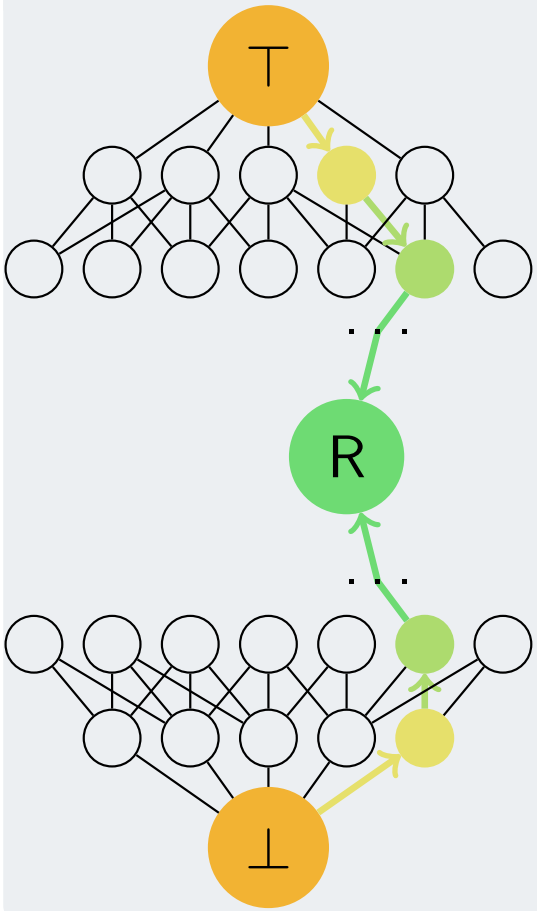
$sat(goodtrain(t^+), 3)$

Layer	Literals
0	$goodtrain(t^+)$
1	$has\_car(t^+, c_1), has\_car(t^+, c_2), has\_car(t^+, c_3)$
2	$wheels(c_1, 2), wheels(c_2, 3), wheels(c_3, 4),$ $long(c_1), long(c_2), long(c_3),$ $load(c_1, circle, 3), load(c_2, circle, 5), load(c_3, triangle, 10),$ $c_1 \neq c_2, c_1 \neq c_3, c_2 \neq c_3$
3	$2 < 3, 2 < 4, 3 < 4, triangle \neq circle, 3 < 5, 3 < 10, 5 < 10$

$vars(sat(goodtrain(V_{t^+}), 3))$

Layer	Literals
0	$goodtrain(V_{t^+})$
1	$has\_car(V_{t^+}, V_{c_1}), has\_car(Vt^+, V_{c_2}), has\_car(V_{t^+}, V_{c_3})$
2	$wheels(V_{c_1}, VW_2), wheels(V_{c_2}, VW_3), wheels(V_{c_3}, VW_4),$ $long(V_{c_1}), long(V_{c_2}), long(V_{c_3}),$ $load(V_{c_1}, Vc, VL_3), load(V_{c_2}, Vc, VL_5), load(V_{c_3}, Vt, VL_10),$ $V_{c_1} \neq V_{c_2}, V_{c_1} \neq V_{c_3}, V_{c_2} \neq V_{c_3}$
3	$VW_2 < VW_3, VW_2 < VW_4, VW_3 < VW_4, Vt \neq Vc,$ $VL_3 < VL_5, VL_3 < VL_{10}, VL_5 < VL_{10}$

Nouvelle approche



- Bi-directionnel et Générer-et-Tester
- Étape de raffinement, un couple d'hypothèses : $(\top_k, \bot_k)$
- $\bot_k = \top_k +$ « saturation »
- $\top_k$ pour construire les raffinements
- $\bot_k$ pour guider la recherche
- fixer les littéraux couche par couche

Ajoute des littéraux

- Ajoute un sous-ensemble de littéraux dans la couche $k+1$ de $\top_k$
- Supprime les littéraux qui ne sont plus atteignables dans $\bot_k$

En pratique, on fait une recherche en largeur parmi les sous-ensembles et on s'arrête quand un est satisfaisant.

Propriétés

- localement fini
- propre
- non-redondant
- pas (faiblement) $\mathcal{L}_\bot$-complet

$\bot_0 = vars(sat(goodtrain(V_{t^+}), 3))$

Layer	Literals
0	$\top_0 = goodtrain(V_{t^+})$
1	$\mathbf{has\_car(V_{t^+}, V_{c_1})}, \mathbf{has\_car(V_{t^+}, V_{c_2})}, \mathbf{has\_car(V_{t^+}, V_{c_3})}$
2	$wheels(V_{c_1}, VW_2), wheels(V_{c_2}, VW_3), wheels(V_{c_3}, VW_4), long(V_{c_1}), long(V_{c_2}), long(V_{c_3}), load(V_{c_1}, Vc, VL_3), load(V_{c_2}, Vc, VL_5), load(V_{c_3}, Vt, VL_10), V_{c_1} \neq V_{c_2}, V_{c_1} \neq V_{c_3}, V_{c_2} \neq V_{c_3}$
3	$VW_2 < VW_3, VW_2 < VW_4, VW_3 < VW_4, Vt \neq Vc, VL_3 < VL_5, VL_3 < VL_10, VL_5 < VL_10$

Example

$\bot_1$?

Layer	Literals
0	$\top_0 = goodtrain(V_{t^+})$
1	$\mathbf{has\_car(V_{t^+}, V_{c_1})}$, $has\_car(V_{t^+}, V_{c_2})$, $has\_car(V_{t^+}, V_{c_3})$
2	$wheels(V_{c_1}, VW_2)$, $wheels(V_{c_2}, VW_3)$, $wheels(V_{c_3}, VW_4)$, $long(V_{c_1})$, $long(V_{c_2})$, $long(V_{c_3})$, $load(V_{c_1}, Vc, VL_3)$, $load(V_{c_2}, Vc, VL_5)$, $load(V_{c_3}, Vt, VL_10)$, $V_{c_1} \neq V_{c_2}, V_{c_1} \neq V_{c_3}, V_{c_2} \neq V_{c_3}$
3	$VW_2 < VW_3, VW_2 < VW_4$, $VW_3 < VW_4, Vt \neq Vc$, $VL_3 < VL_5, VL_3 < VL_10, VL_5 < VL_10$

$\top_1$

$\bot_1$

$\Rightarrow$ Couvre t^-

Example

$\bot_1$?

Layer	Literals	
0	$\top_0 = goodtrain(V_{t^+})$	$\top_1$
1	$\mathbf{has\_car(V_{t^+}, V_{c_1})}, \mathbf{has\_car(V_{t^+}, V_{c_2})}$, $has\_car(V_{t^+}, V_{c_3})$	
2	$wheels(V_{c_1}, VW_2), wheels(V_{c_2}, VW_3)$, $wheels(V_{c_3}, VW_4), long(V_{c_1})$, $long(V_{c_2}), long(V_{c_3})$, $load(V_{c_1} Vc, VL_3), load(V_{c_2}, Vc, VL_5)$, $load(V_{c_3}, Vt, VL_10)$, $V_{c_1} \neq V_{c_2}, V_{c_1} \neq V_{c_3}, V_{c_2} \neq V_{c_3}$	$\bot_1$
3	$VW_2 < VW_3, VW_2 < VW_4$, $VW_3 < VW_4, Vt \neq Vc$, $VL_3 < VL_5, VL_3 < VL_10, VL_5 < VL_10$	

$\Rightarrow$ Rejette t^-

Evaluation

benchmark	Propal			Aleph2		
	# learned rules	time (s)	acc.	# learned rules	time (s)	acc.
Graph coloring	1	0	100%	1	0.14	100%
School timetable	3	11	98,33%	1	0.31	100%
Job-shop	6	103	87,78%	6	1130.88	96%
N-queens	-	-	-	3	560.76	61.67%
	Progolem			Aleph1		
Graph coloring	1	0.18	100%	1	0.24	100%
School timetable	1	11.91	100%	1	1.24	100%
Job-shop	-	-	-	3	1051.03	100%
N-queens	-	-	-	3	489.49	100%

benchmark	Bidirectional		
	# learned rules	time (s)	acc.
Graph coloring	1	0.17	100%
School timetable	2	0.69	100%
Job-shop	5	7.37	100%
N-queens	3	29.11	100%

benchmark	Propal # learned rules	Propal time (s)	Propal acc.	Aleph2 # learned rules	Aleph2 time (s)	Aleph2 acc.
Graph coloring	1	0	100%	1	0.14	100%
School timetable	3	11	98,33%	1	0.31	100%
Job-shop	6	103	87,78%	6	1130.88	96%
N-queens	-	-	-	3	560.76	61.67%
	Progolem			Aleph1		
Graph coloring	1	0.18	100%	1	0.24	100%
School timetable	1	11.91	100%	1	1.24	100%
Job-shop	-	-	-	3	1051.03	100%
N-queens	-	-	-	3	489.49	100%

benchmark	Bidirectional # learned rules	Bidirectional time (s)	Bidirectional acc.
Graph coloring	1	0.17	100%
School timetable	2	0.69	100%
Job-shop	5	7.37	100%
N-queens	3	29.11	100%

Limites :

- approche conçue pour des exemples structurés en couches
- exploration des sous-ensembles doit s'arrêter avant qu'elle soit complète

Pour résumer

- Nouvel opérateur de raffinement en ILP
- Bi-directionnel et générer-et-tester
- Basé sur la structure en couches induite par la saturation
- Fixer couche par couche les littéraux de la règle

Summary

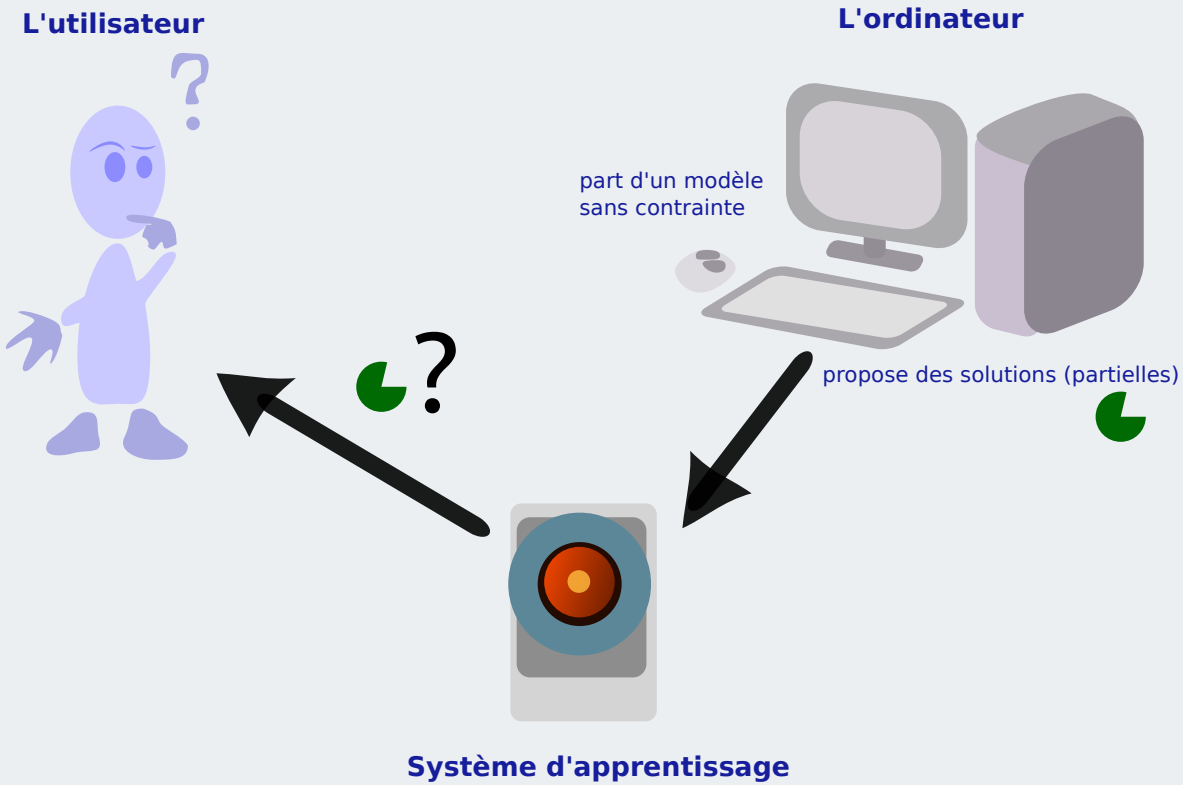
1 Apprentissage de modèles abstraits

2 Générer-et-Tester et opérateur bi-directionnel

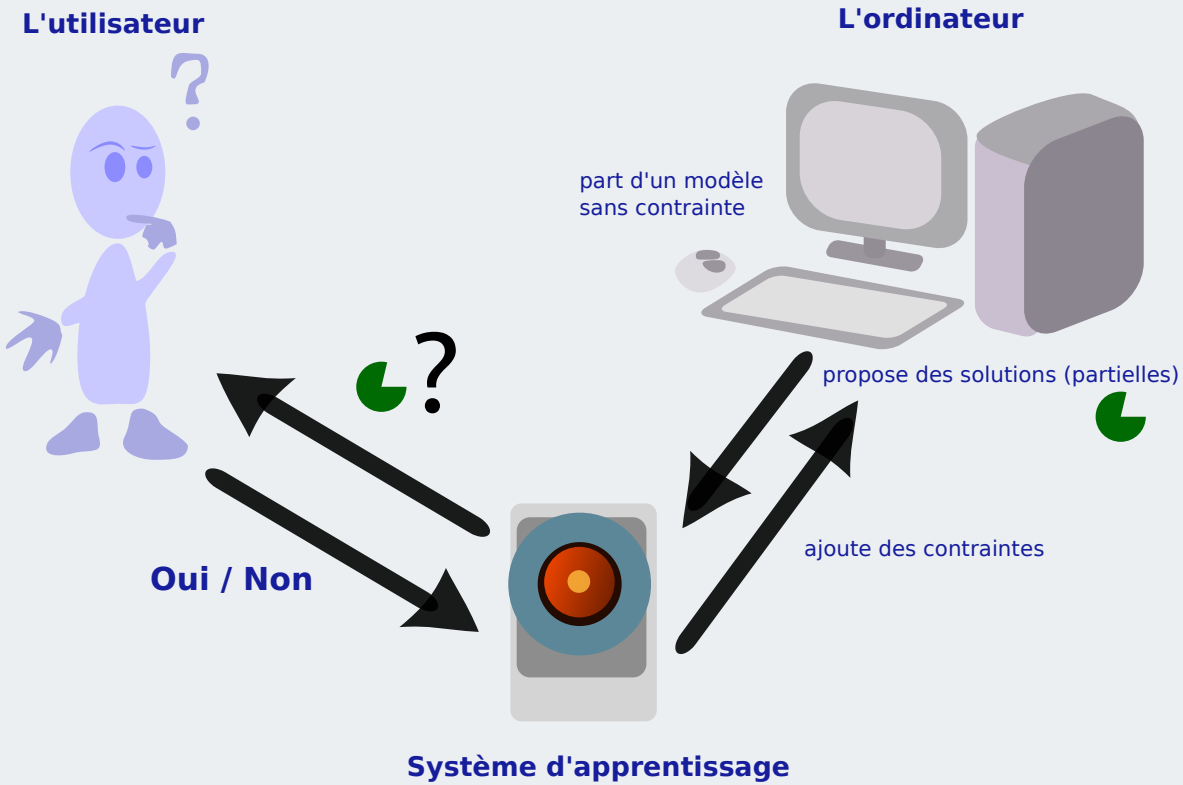
3 Recherche Interactive

4 Conclusion

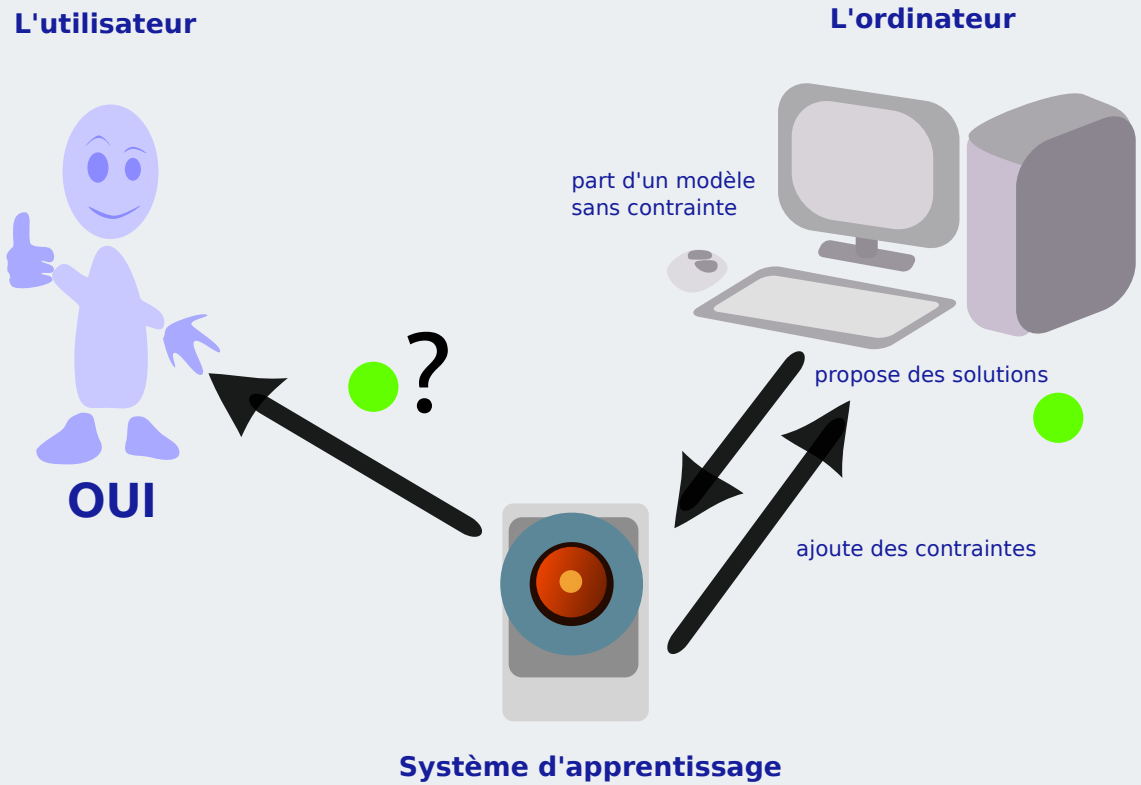
Recherche interactive



Recherche interactive

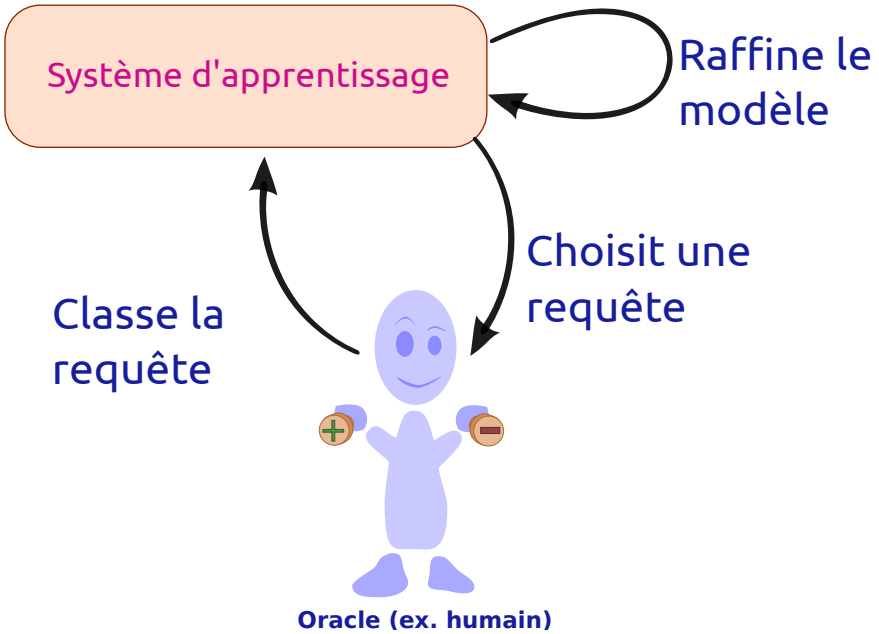


Recherche interactive



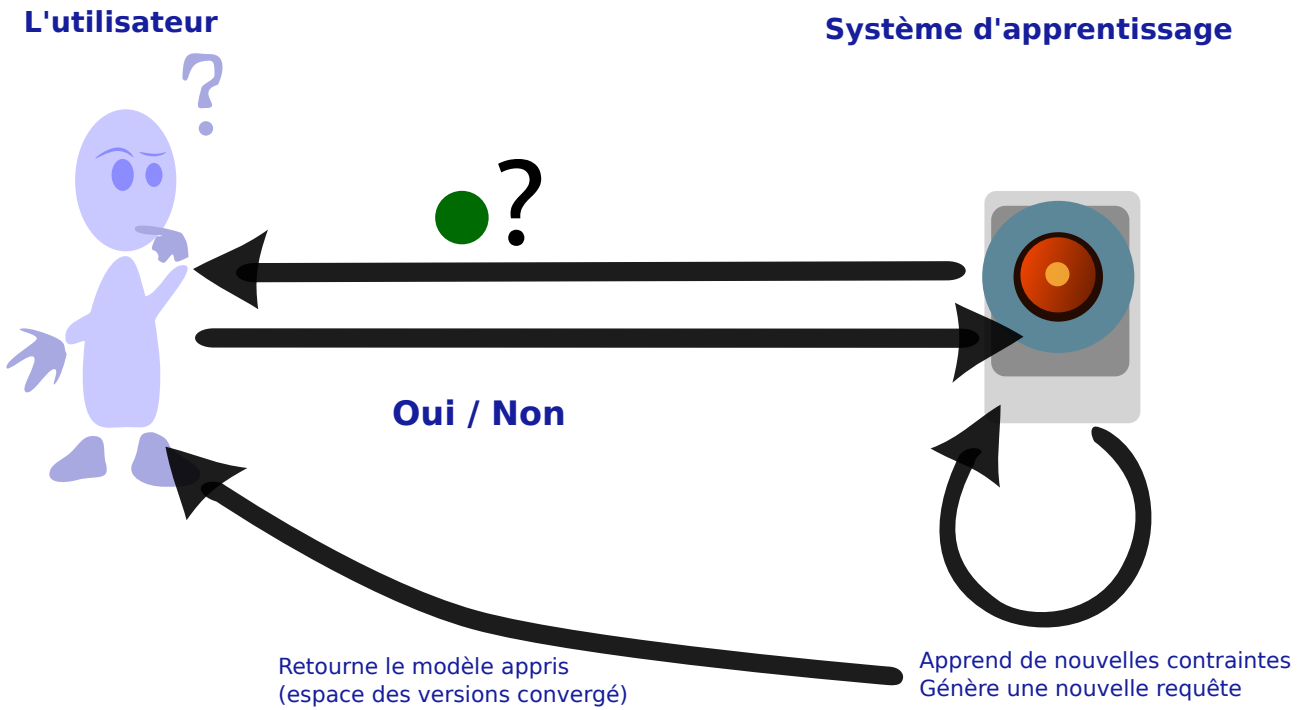
Les questions sont des requêtes.

Dans notre cas, des affectations des variables du CSP



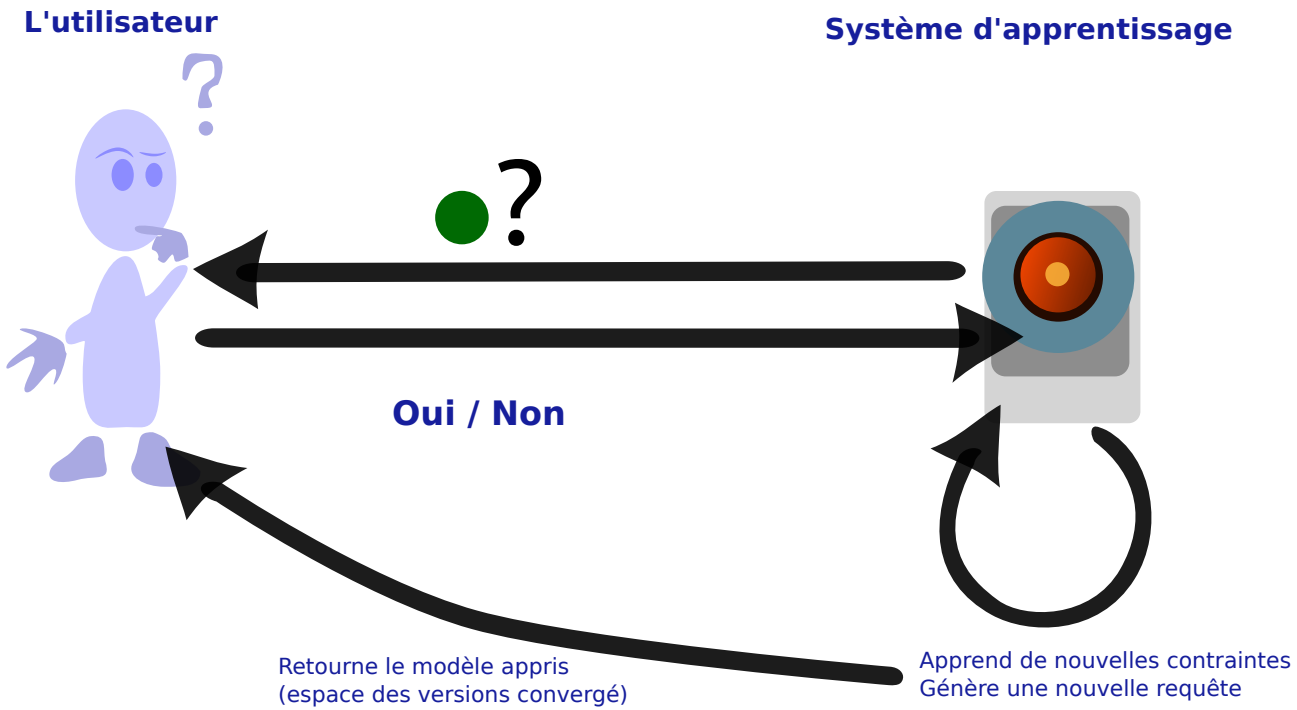
Dans notre cas, on s'arrêtera quand on aura une solution du CSP

CONACQ dirigé par les requêtes [Bessière et al 2007]



- Pas le même objectif (apprendre le modèle)
- Besoin de demander à l'utilisateur de classer des solutions

CONACQ dirigé par les requêtes [Bessière et al 2007]



- Pas le même objectif (apprendre le modèle)
- Besoin de demander à l'utilisateur de classer des solutions

Introduction de requêtes partielles

Exemple :

Variables : X_0, X_1, X_2, X_3

Domaine : $\{1, 2, 3, 4\}$

Type de contraintes : $\leq, \neq, \geq$

Requête complète

Affectation des variables : $(X_0/1, X_1/2, X_2/3, X_3/4)$

Introduction de requêtes partielles

Exemple :

Variables : X_0, X_1, X_2, X_3

Domaine : $\{1, 2, 3, 4\}$

Type de contraintes : $\leq, \neq, \geq$

Requête partielle

Affectation partielle des variables : $(X_0/1, X_1/\underline{?}, X_2/3, X_3/4)$

Introduction de requêtes partielles

Exemple :

Variables : X_0, X_1, X_2, X_3

Domaine : $\{1, 2, 3, 4\}$

Type de contraintes : $\leq, \neq, \geq$

Requête partielle

Affectation partielle des variables : $(X_0/1, X_1/\underline{?}, X_2/3, X_3/4)$

Réponse OUI : la requête ne viole pas de contrainte

Réponse NON : la requête viole au moins une contrainte

Qu'apprend-on avec une requête ?

Exemple :

Type de contraintes : $\leq, \neq, \geq$

Requête : $(X_0/1, X_1/?, X_2/3, X_3/4)$

Contraintes violées

$\{X_0 \geq X_3, X_0 \geq X_4, X_3 \geq X_4\}$

Requête positive

Les contraintes violées $\Rightarrow$ pas dans le modèle (ignorées ensuites)

Requête négative

Au moins, une des contraintes violées est dans le modèle

Ajout de la contrainte $X_0 \geq X_3 \vee X_0 \geq X_4 \vee X_3 \geq X_4$

Qu'apprend-on avec une requête ?

Exemple :

Type de contraintes : $\leq, \neq, \geq$

Requête : $(X_0/1, X_1/?, X_2/3, X_3/4)$

Contraintes violées

$\{X_0 \geq X_3, X_0 \geq X_4, X_3 \geq X_4\}$

Requête positive

Les contraintes violées $\Rightarrow$ pas dans le modèle (ignorées ensuites)

Requête négative

Au moins, une des contraintes violées est dans le modèle

Ajout de la contrainte $X_0 \geq X_3 \vee X_0 \geq X_4 \vee X_3 \geq X_4$

Qu'apprend-on avec une requête ?

Exemple :

Type de contraintes : $\leq, \neq, \geq$

Requête : $(X_0/1, X_1/?, X_2/3, X_3/4)$

Contraintes violées

$\{X_0 \geq X_3, X_0 \geq X_4, X_3 \geq X_4\}$

Requête positive

Les contraintes violées $\Rightarrow$ pas dans le modèle (ignorées ensuites)

Requête négative

Au moins, une des contraintes violées est dans le modèle
Ajout de la contrainte $X_0 \geq X_3 \vee X_0 \geq X_4 \vee X_3 \geq X_4$

Basé sur l'arbre de résolution

Arbre de recherche

- à chaque nœud : affecte une valeur à une variable
- exploration en profondeur d'abord
- les contraintes permettent d'éliminer des valeurs des domaines (et donc des branches)

Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{1,2,3\}$
- $D_Y : \{1,2,3\}$
- $D_Z : \{1,2,3\}$

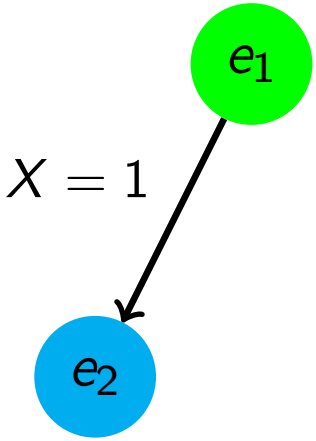
e_1

Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{1,2\}$
- $D_Y : \{2,3\}$
- $D_Z : \{1,2,3\}$

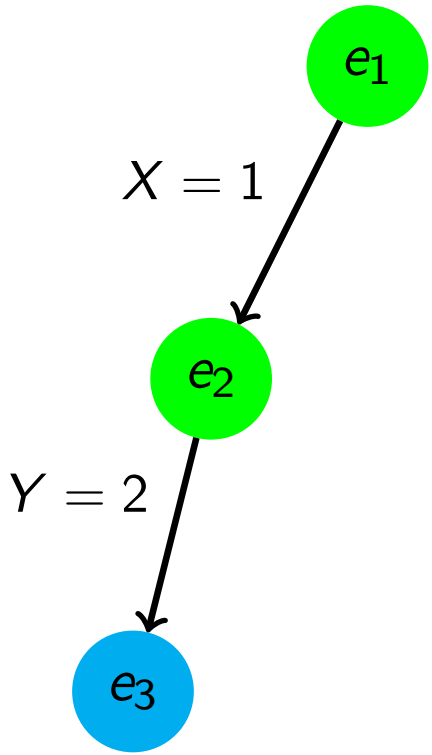


Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{1\}$
- $D_Y : \{2,3\}$
- $D_Z : \{1,2,3\}$

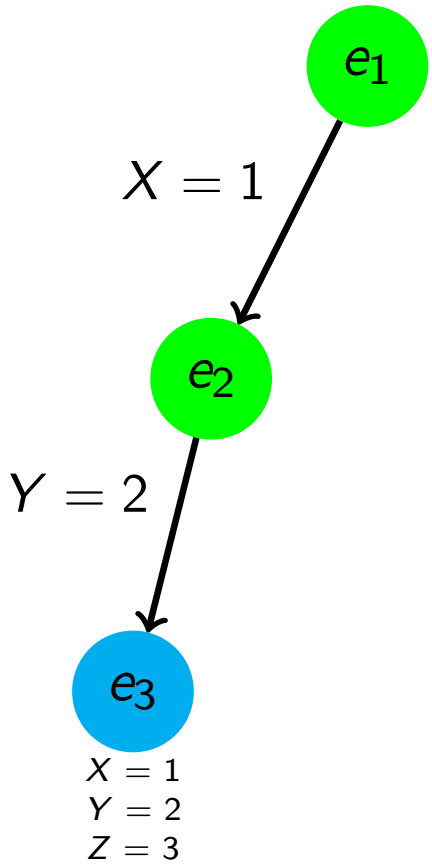


Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{1\}$
- $D_Y : \{2\}$
- $D_Z : \{2,3\}$

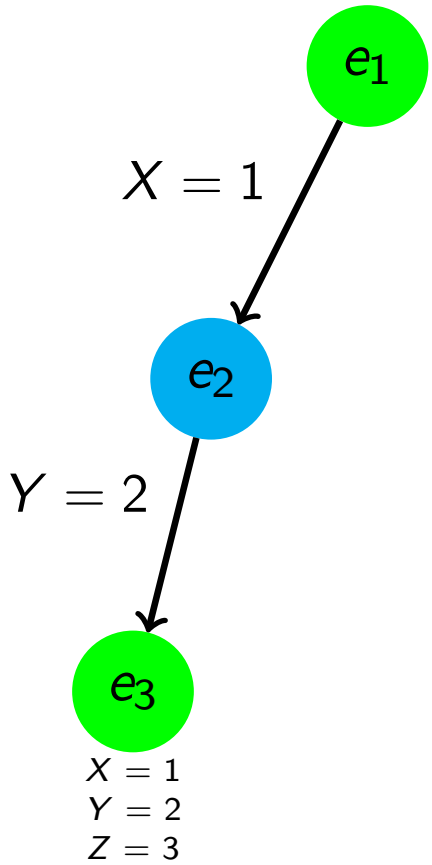


Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{1\}$
- $D_Y : \{2\}$
- $D_Z : \{3\}$

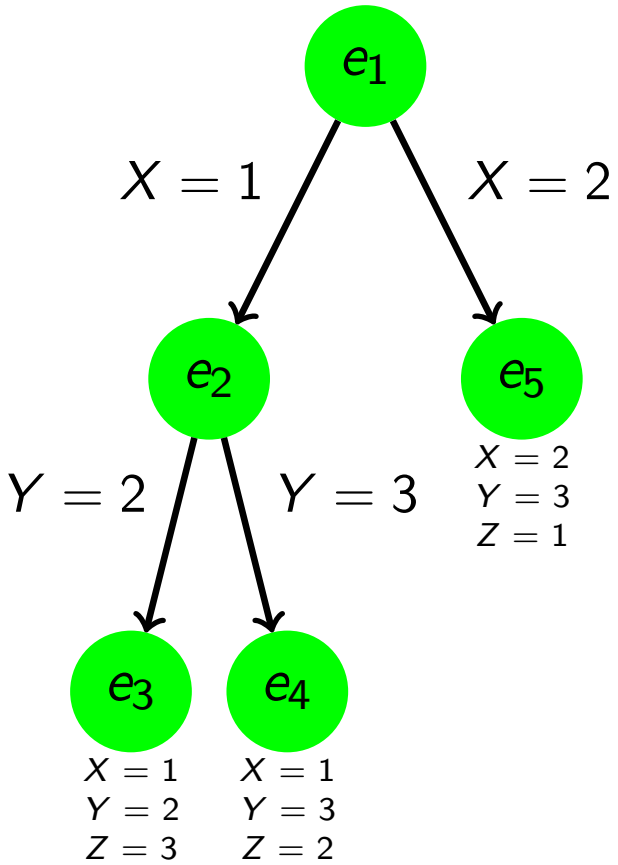


Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{1\}$
- $D_Y : \{2,3\}$
- $D_Z : \{2,3\}$



Contraintes

- $X < Y$
- $X \neq Z$
- $Y \neq Z$

Domaines

- $D_X : \{2\}$
- $D_Y : \{3\}$
- $D_Z : \{1\}$

Notre générateur : *Branch-&-Ask*

Pour chaque nœud,

- soit le solveur peut décider avec ce qu'il connaît du modèle (requête redondante)
- soit une requête est posée à l'utilisateur

Exemple

Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

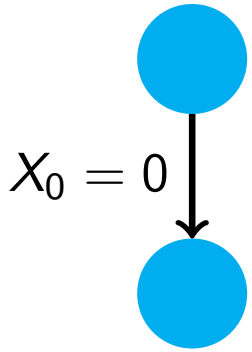
Modèle

Contraintes ignorées

Nœud courant

Contraintes violées : $\{\}$

Exemple



Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

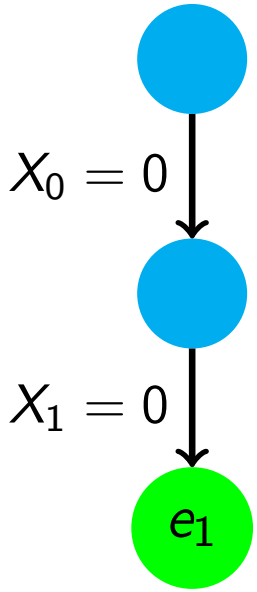
Modèle

Contraintes ignorées

Nœud courant

Contraintes violées : $\{\}$

Exemple



Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

Modèle

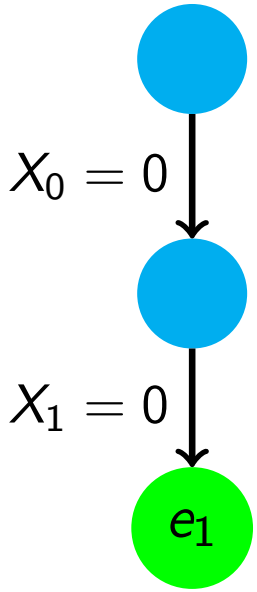
Contraintes ignorées

Nœud courant

Contraintes violées : $\{X_0 \neq X_1\}$

Requête !

Exemple



Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

Modèle

Contraintes ignorées

$X_0 \neq X_1$

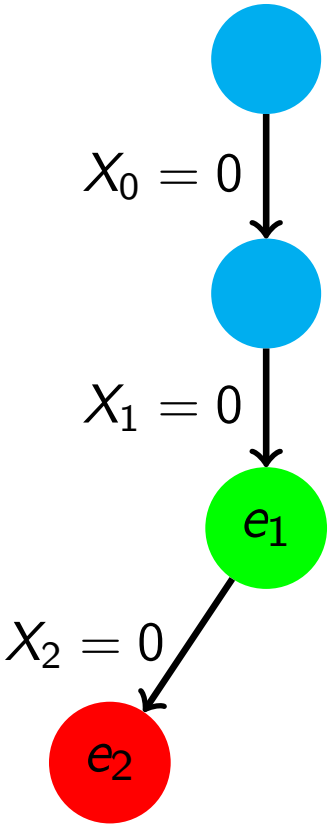
Nœud courant

Contraintes violées :

$\{X_0 \neq X_1\}$

Requête !

Exemple



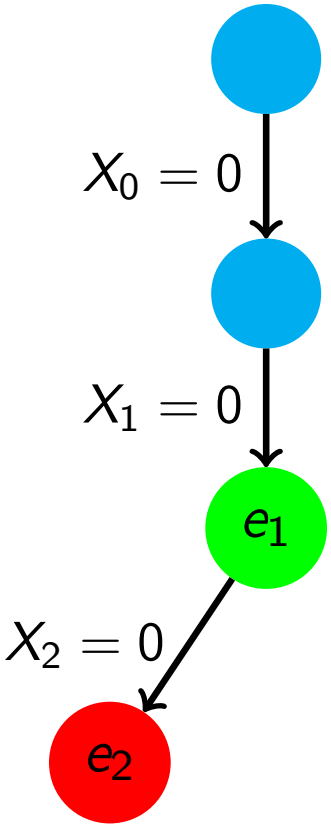
Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

Modèle

Contraintes ignorées
$X_0 \neq X_1$

Nœud courant
Contraintes violées :
$\{X_0 \neq X_2, X_1 \neq X_2\}$
Requête !

Exemple



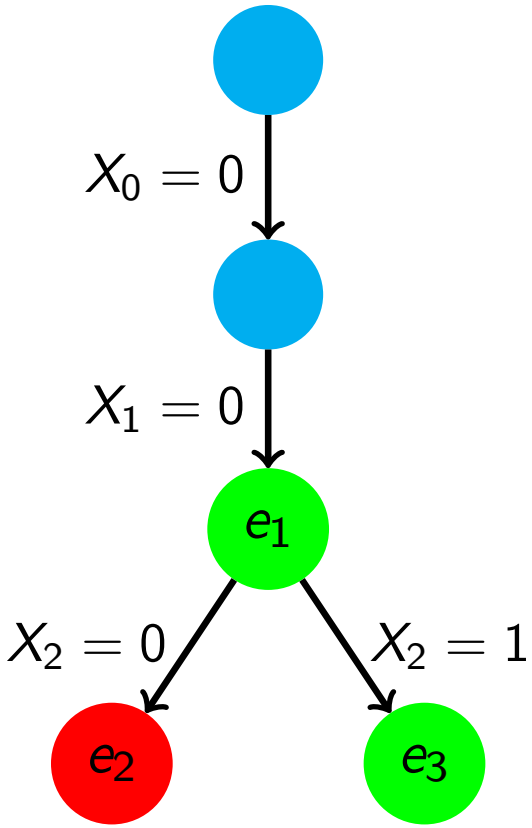
Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

Modèle
$X_0 \neq X_2 \vee X_1 \neq X_2$

Contraintes ignorées
$X_0 \neq X_1$

Nœud courant
Contraintes violées :
$\{X_0 \neq X_2, X_1 \neq X_2\}$
Requête !

Exemple



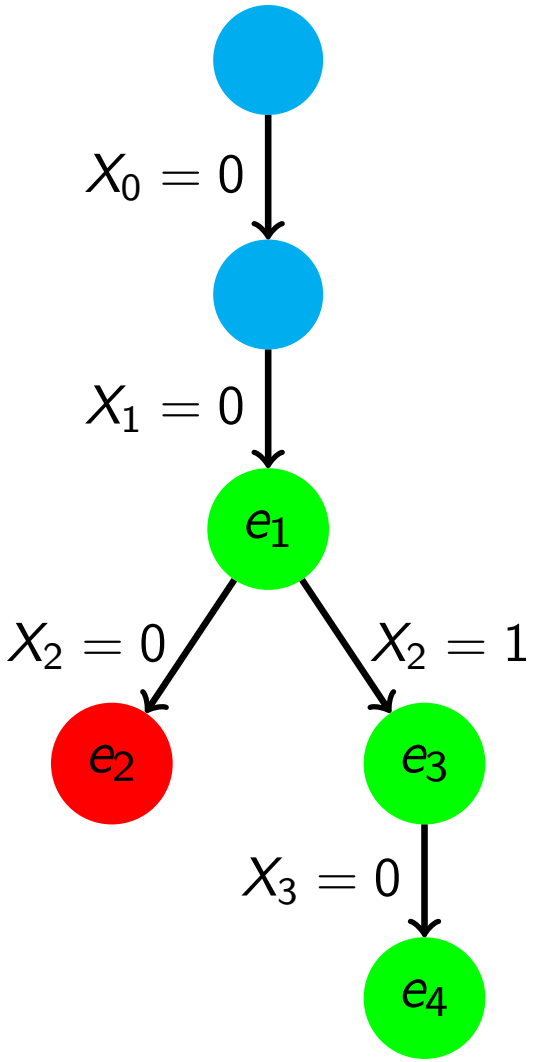
Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

Modèle
$X_0 \neq X_2 \vee X_1 \neq X_2$

Contraintes ignorées
$X_0 \neq X_1$, $X_0 \geq X_2$, $X_1 \geq X_2$

Nœud courant
Contraintes violées :
$\{X_0 \geq X_2, X_1 \geq X_2\}$
Requête !

Exemple



Contraintes cibles : $X_0 \leq X_3$, $X_1 \neq X_2$, $X_2 \geq X_3$

Modèle

$X_0 \neq X_2 \vee X_1 \neq X_2$

Contraintes ignorées

$X_0 \neq X_1$, $X_0 \geq X_2$, $X_1 \geq X_2$

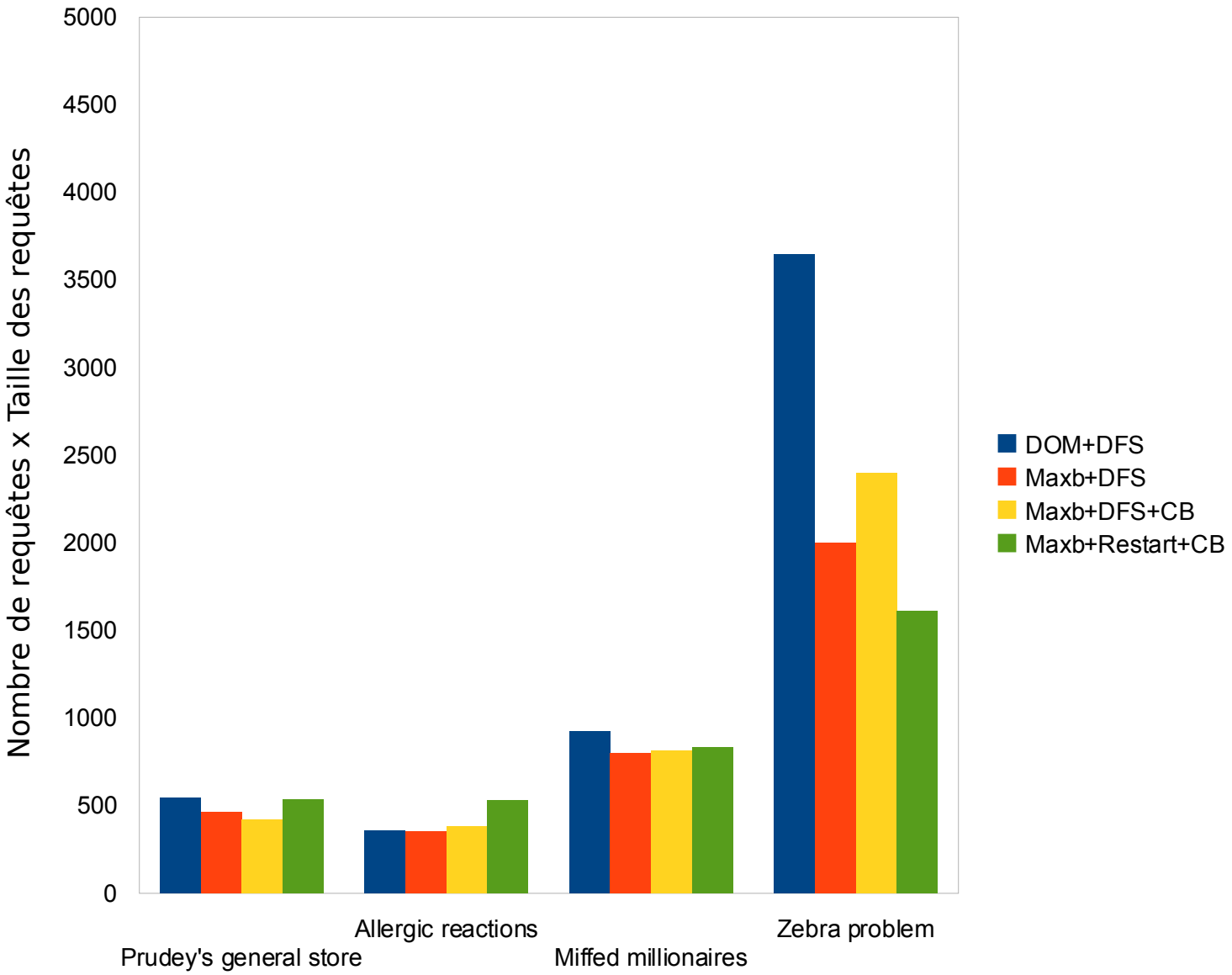
Nœud courant

Contraintes violées :

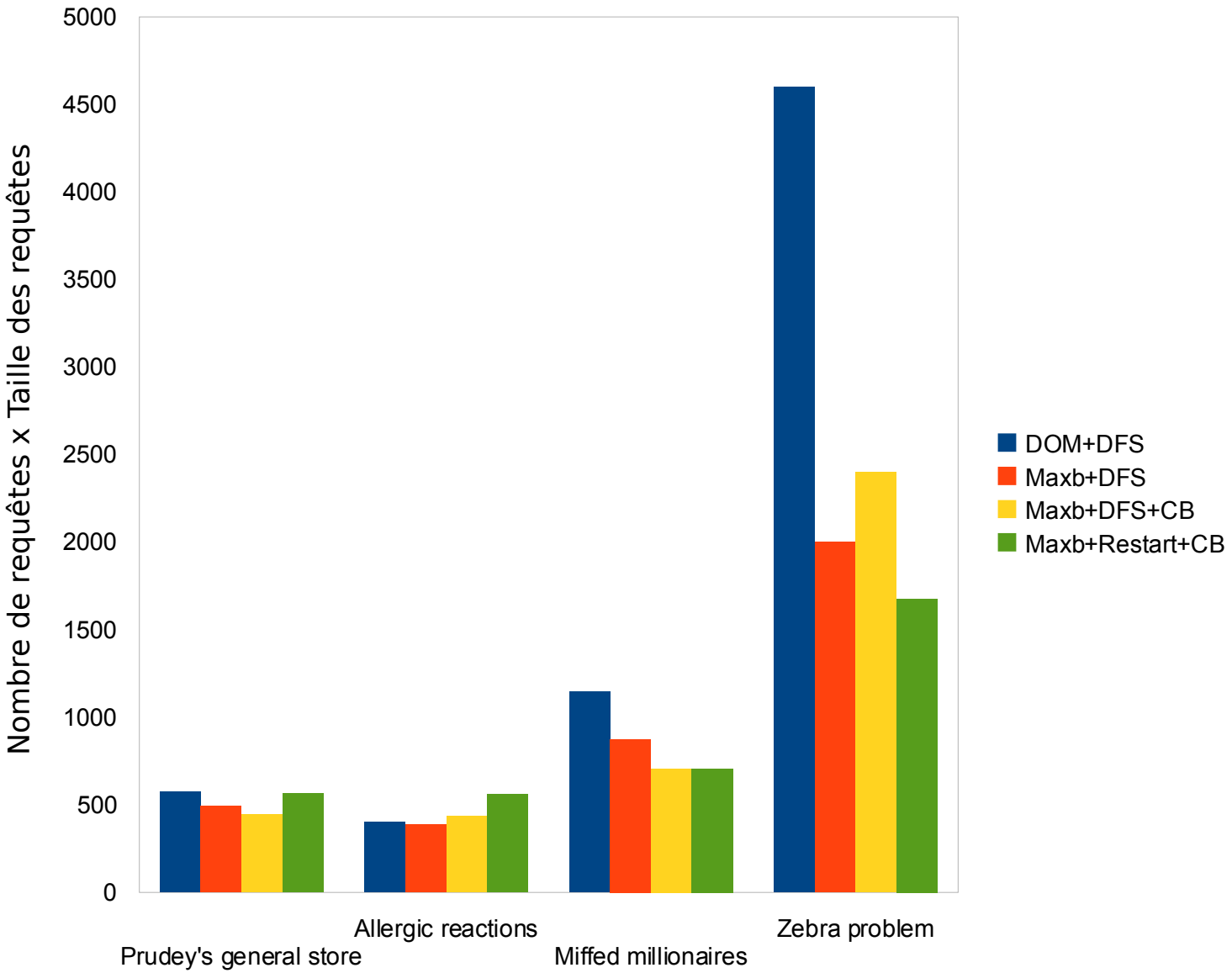
$\{X_0 \neq X_3, X_2 \geq X_3, X_1 \neq X_3,\}$

Requête ! Solution !

Évaluation — 1 solution



Évaluation — Toutes les solutions



Pour résumer

- Changement de problématique : chercher une solution vs chercher un modèle
- Cadre de l'apprentissage actif (extension de CONACQ dirigé par les requêtes)
- Introduction des requêtes partielles
- Générateur de requêtes biaisé par l'arbre de résolution du CSP
- Extensible à l'apprentissage de modèle (recherche de toutes les solutions)
- Évaluation sur des problèmes avec peu de solutions encourageante !

Summary

1 Apprentissage de modèles abstraits

2 Générer-et-Tester et opérateur bi-directionnel

3 Recherche Interactive

4 Conclusion

Pour résumer

Une thèse à mi-chemin entre la programmation par contraintes et l'apprentissage automatique.

- Thème central : simplifier l'étape de modélisation
- Refonte des problématiques

  $\Rightarrow$ 2 nouveaux axes de recherche

1. **Apprentissage de modèles abstraits**
   - Un cadre permettant d'obtenir un réseau de contraintes en passant par un modèle abstrait
   - Un nouveau système d'apprentissage basé sur la structure du modèle abstrait

   \+ Utiliser des solutions/non-solutions de problèmes proches
   \+ Apprendre un modèle plus haut-niveau
   − Complexité de l'approche (espace de recherche et plateaux)

Pour résumer

1. **Apprentissage de modèles abstraits**
   - Un cadre permettant d'obtenir un réseau de contraintes en passant par un modèle abstrait
   - Un nouveau système d'apprentissage basé sur la structure du modèle abstrait
2. **Recherche interactive**
   - Apprentissage actif
   - Utilisation de requêtes partielles
   - Générateur biaisé par l'arbre de résolution

\+ Approche réaliste pour l'utilisateur
\+ Modularité (possibilité d'avoir d'autres stratégies)
− Nombre de requêtes encore important

Apprentissage de modèles abstraits

- Augmenter l'expressivité du langage mi-niveau (aggregat)
- Générer des exemples négatifs
  - Des exemples plus petits, plus ciblés
- Voir du côté de la propositionnalisation
- Opérateur de raffinement
  - Impact des couches sur l'efficacité
  - Générer des modèles abstraits aléatoirement
  - Lien avec la transition de phases en ILP
  - Exploiter la structure des exemples ?
  - « Familles » de problèmes d'apprentissage ?

Recherche interactive

- Meilleure compréhension des phénomènes
- Meilleure heuristique
- Règles de redondance ?
- Générateur non biaisé par l'arbre de recherche
- D'autres types de requêtes

Problème du zèbre

Strategie	Dom			Maxb		
	$\#q$	$\|q\|$	$m(q)$	$\#q$	$\|q\|$	$m(q)$
DFS						
1 solution	132	7	924	160	5	**800**
toutes	164	6	1148	175	5	875
DFS+clauses breaking						
1 solution	143	7	1001	163	5	815
toutes	178	6	1002	177	4	**708**
Restart+clauses breaking						
1 solution	144	7	1008	166	5	830
toutes	178	6	1068	177	4	**708**