Elimination des symétries dans les problèmes injectifs

Jean-François Puget

Elimination des symétries dans les problèmes injectifs

Jean-François Puget
ILOG
9 Avenue de Verdun
94253 Gentilly Cedex
puget@ilog.fr

Résumé

L'ajout de contraintes est utilisé depuis longtemps pour éliminer les symétries dans les CSP. Par exemple, on peut enlever toutes les symétries dans le problème du 'pigeon hole" en ordonnant les variables. Nous avons généralisé ce résultat à tous les problèmes injectifs, c'est-à-dire où les variables doivent prendre des valeurs toutes distinctes. Dans ce cas il est possible d'éliminer toutes les symétries avec un ordre partiel sur les variables. Nous montrons comment automatiquement calculer un tel ordre en utilisant des algorithmes sur les groupes finis. Nous montrons également que cet ordre partiel peut être combiné avec une méthode éliminant les symétries de valeurs. Des expériences variées montrent l'intérêt pratique de notre méthode.

Abstract

Adding symmetry breaking constraints is one of the oldest ways of breaking variable symmetries for CSPs. For instance, it is well known that all the symmetries for the pigeon hole problem can be removed by ordering the variables. We have generalized this result to all CSPs where the variables are subject to an all different constraint. In such case it is possible to remove all variable symmetries with a partial ordering of the variables. We show how this partial ordering can be automatically computed using computational group theory (CGT). We further show that partial orders can be safely used together with the GE-tree method of [10]. Experiments show the efficiency of our method.

1 Introduction

Une symétrie pour un problème de satisfaction de contraintes (CSP, Constraint Satisfaction Problem) est une fonction du CSP sur lui même qui préserve sa structure et ses solutions. Si un CSP a des symétries, il se peut que toutes les variantes symétriques d'un échec soient testés avant de trouver une solution. Même si le problème est facile à résoudre, toute variante symétrique d'une solution est une solution, et il peut être impossible de les lister.

L'ajout de contraintes est une des méthodes les plus anciennes pour éliminer des symétries [9]. Par exemple, il est montré dans [1] qu'il est possible d'éliminer toutes les symétries de variables en ajoutant une contrainte lexicographique par symétrie. Malheureusement, cette méthode n'est pas pratique car il peut y avoir un nombre exponentiel de symétries. Il est d'ailleurs montré dans [11] qu'il n'est pas possible en général d'éliminer toutes les symétries avec un nombre polynomial de contraintes. Dans [2], un nombre linéaire de contraintes est utilisé pour éliminer les symétries dans des problèmes pouvant se représenter par une matrice de variables. Comme le nombre de contraintes ajoutées est polynomial, certaines symétries ne sont pas éliminées. Toutefois, dans certains cas particuliers, il peut être possible d'éliminer toutes les symétries avec un nombre polynomial de contraintes. Par exemple, dans [9], nous montrons que le problème du "pigeon hole" peut être facilement résolu en ordonnant toutes les variables.

Dans cet article nous considérons une classe de problèmes plus générale : les problèmes injectifs. Il s'agit de problèmes où les variables doivent prendre des valeurs différentes deux à deux, par exemple parce qu'une contrainte "tous différents" est présente dans le problème. Dans ce cas, la fonction ayant pour domaine les variables et pour image les valeurs est injective. Nous montrons dans la section 3 que dans ce cas toutes les symétries de variables peuvent être éliminées

avec $n-1$ contraintes binaires, n étant le nombre de variables.

Dans [10] une méthode générale pour éliminer toutes les symétries de valeurs est présentée. Nous montrons dans la section 4 comment combiner cette méthode avec les contraintes éliminant les symétries de variables.

Dans la section 5 nous appliquons notre méthode à quelques problèmes complexes, et nous concluons dans la section 6.

2 Symétries, Graphes et CSPs

Les symétries que nous considérons sont des permutations, c'est-à-dire des bijections d'un ensemble fini sur lui même. Sans perte de généralité nous pouvons nous restreindre à l'étude des permutations de I^n, avec $I^n = \{0, 1, 2, \ldots, n-1\}$. Par exemple, nous pouvons indexer les variables d'un CSP avec des entiers, de façon à ce que toute symétrie de variables soit définie par une permutation des indices des variables. Ceci est formalisé comme suit.

2.1 Algorithmes de théorie des groupes

Soit S^n l'ensemble des permutations de I^n. L'image par i par la permutation σ est notée i^σ. Une permutation $\sigma \in S^n$ est entièrement décrite par le vecteur $[0^\sigma, 1^\sigma, \ldots, (n-1)^\sigma]$. La composition de deux permutations σ et θ est définie par $i^{(\sigma\theta)} = (i^\sigma)^\theta$.

Etant donné $i \in I^n$ et un groupe de permutations $G \subseteq S^n$, l'*orbite* de i dans G, notée i^G, est l'ensemble des images de i par les éléments de G :

$$i^G = \{i^\sigma | \sigma \in G\}$$

Etant donné $i \in I^n$ et un groupe de permutations $G \subseteq S^n$, le *stabilisateur* de i dans G, noté i_G, est l'ensemble des éléments de G qui laissent i invariant :

$$i_G = \{\sigma \in G | i^\sigma = i\}$$

2.2 CSP et symétries

Un CSP $\mathcal{P}$ avec n variables est un triplet $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ où $\mathcal{V}$ est un ensemble fini de variables $(v_i)_{i \in I^n}$, $\mathcal{D}$ un ensemble fini d'ensembles finis $(\mathcal{D}_i)_{i \in I^n}$, et chaque contrainte de $\mathcal{C}$ un sous ensemble du produit cartésien $\bigotimes_{i \in I^n} \mathcal{D}_i$. Sans perte de généralité, nous pouvons supposer que $\mathcal{D}_i \subseteq I^k$ pour k donné k.

Un *assignement* est un vecteur de valeurs $(a_i)_{i \in I^n}$ tel que $a_i \in \mathcal{D}_i$ pour tout $i \in I^n$, et est noté $(v_i = a_i)_{i \in I^n}$. Un *assignement partiel* est un sous vecteur d'un assignement.

Une *solution* de $(\mathcal{V}, \mathcal{D}, \mathcal{C})$ est un assignement qui satisfait toutes les contraintes.

Etant donné une permutation σ de I^n, on définit une permutation de variables sur les assignements (partiels ou non) par :

$$((v_i = a_i)_{i \in I^n})^\sigma = ((v_{i^\sigma} = a_i)_{i \in I^n})$$

De telles permutations sont des symétries de variables si elles envoient les solutions sur des solutions.

Etant donné une permutation θ de I^k, on définit une permutation de valeurs sur les assignements (partiels ou non) par :

$$((v_i = a_i)_{i \in I^n})^\theta = ((v_i = a_i^{\theta^{-1}})_{i \in I^n})$$

De telles permutations sont des symétries de valeurs si elles envoient les solutions sur des solutions.

2.3 Un coloriage de graphes

L'exemple qui suit va être utilisé pour illustrer notre méthode. Un graphe avec m arêtes est dit *gracieux* s'il existe un étiquetage f de ses sommets tel que :

- $0 \leq f(i) \leq m$ pour tout sommet i,
- f est injective,
- les valeurs $abs(f(i), f(j))$ pour toutes les arêtes (i, j) sont toutes différentes.

Ceci peut se modéliser naturellement comme un CSP avec une variable v_i par sommet [4]. Les symétries du problème sont induites par les automorphismes du graphe. Il y a une symétrie de valeurs non triviale, qui envoie v sur $m - v$. Pour plus d'information sur les symétries des graphes gracieux voir [7], [8].

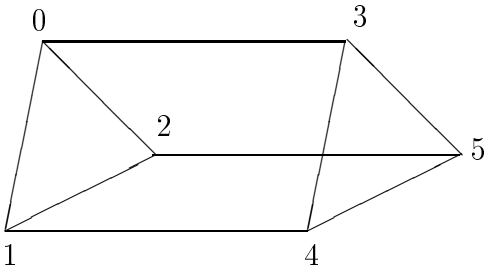

Figure 1. Le graphe $K3 \times P2$.

Considérons le graphe de la figure 1. Le groupe des symétries de variables pour le CSP correspondant est équivalent au groupe des symétries du graphe. Un tel groupe peut être calculé avec un logiciel tel que Nauty[5]. Ce groupe G est :

$$\{[0,1,2,3,4,5], [0,2,1,3,5,4], [1,0,2,4,3,5],$$
$$[1,2,0,4,5,3], [2,0,1,5,3,4], [2,1,0,5,4,3],$$
$$[3,4,5,0,1,2], [3,5,4,0,2,1], [4,3,5,1,0,2],$$
$$[4,5,3,1,2,0], [5,3,4,2,0,1], [5,4,3,2,1,0]\}$$

3 Elimination des symétries de variables

Comme nous l'avons dit, l'ajout de contraintes est une méthode connue pour éliminer les symétries de variables.

3.1 Contraintes lexicographiques

Dans [1], il est montré que toutes les symétries de variables pouvaient être éliminées avec les contraintes suivantes.

$$\forall \sigma \ \in \ G, \ \mathcal{V} \preceq \mathcal{V}^{\sigma} \tag{1}$$

Pour une symétrie σ donnée, la contrainte $(\mathcal{V} \preceq \mathcal{V}^{\sigma})$ est équivalent à la disjonction des contraintes :

$$\begin{array}{c} v_0 < v_{0^\sigma} \\ v_0 = v_{0^\sigma} \wedge v_1 < v_{1^\sigma} \\ \vdots \\ v_0 = v_{0^\sigma} \wedge \ldots \wedge v_{i-1} = v_{(i-1)^\sigma} \wedge v_i < v_{i^\sigma} \\ \vdots \\ v_0 = v_{0^\sigma} \wedge \ldots \wedge v_{n-2} = v_{(n-2)^\sigma} \wedge v_{n-1} < v_{(n-1)^\sigma} \\ v_0 = v_{0^\sigma} \wedge \ldots \wedge v_{n-2} = v_{(n-2)^\sigma} \wedge v_{n-1} = v_{(n-1)^\sigma} \end{array}$$

Si la dernière contrainte est omise, l'ensemble des contraintes est noté $\mathcal{V} \prec \mathcal{V}^{\sigma}$.

Dans notre exemple, les contraintes données par [1] sont :

$$\begin{array}{lcl} (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_0, v_1, v_2, v_3, v_4, v_5) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_0, v_2, v_1, v_3, v_5, v_4) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_1, v_0, v_2, v_4, v_3, v_5) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_1, v_2, v_0, v_4, v_5, v_3) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_2, v_0, v_1, v_5, v_3, v_4) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_2, v_1, v_0, v_5, v_4, v_3) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_3, v_4, v_5, v_0, v_1, v_2) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_3, v_5, v_4, v_0, v_2, v_1) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_4, v_3, v_5, v_1, v_0, v_2) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_4, v_5, v_3, v_1, v_2, v_0) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_5, v_3, v_4, v_2, v_0, v_1) \\ (v_0, v_1, v_2, v_3, v_4, v_5) & \preceq & (v_5, v_4, v_3, v_2, v_1, v_0) \end{array}$$

3.2 Un nombre polynomial de contraintes

Le nombre de contraintes (1) peut être exponentiel avec le nombre de variables $\mathcal{V}$. Ce nombre peut être grandement réduit en utilisant l'utilisation de l'injectivité de la fonction des variable vers les valeurs. Prenons une des symétries de notre exemple :

$$\sigma = [0, 2, 1, 3, 5, 4]$$

La contrainte éliminant cette symétrie est :

$$(v_0, v_1, v_2, v_3, v_4, v_5) \preceq (v_0, v_2, v_1, v_3, v_5, v_4)$$

Puisque $v_0 = v_0$ est trivialement vraie, et puisque $v_1 = v_2$ ne peut pas être vrai par injectivité, cette contrainte peut se simplifier en :

$$v_1 < v_2$$

Cette simplification est vraie en général, et peut être formalisée comme suit. Etant donné une permutation σ, notons $s(\sigma)$ le plus petit i tel que $i^\sigma \neq i$, et notons $t(\sigma)$ l'image de $(s(\sigma))$ par σ.

Lemme 1. *Etant donné un CSP (V,D,C) injectif, et un groupe de symétries de variables G pour le CSP, alors toutes les symétries de G peuvent être éliminées par les contraintes suivantes :*

$$\forall \sigma \in G, v_{s(\sigma)} < v_{t(\sigma)} \tag{2}$$

Preuve. Par définition, $k^\sigma = k$ si $k < s(\sigma)$, et $s(\sigma)^\sigma \neq s(\sigma)$. Examinons la contrainte $\mathcal{V} \preceq \mathcal{V}_\sigma$. L'injectivité du CSP implique $v_i = v_{i^\sigma}$ si et seulement si $i^\sigma = i$. En particulier, $v_k = v_{k^\sigma}$ for all $k < s(\sigma)$, et $v_{s(\sigma)} \neq v_{(s(\sigma))^\sigma}$. Donc un seul élément de la disjonction peut être vrai. Il s'agit de :

$$v_0 = v_{0^\sigma} \wedge \ldots \wedge v_{s(\sigma)-1} = v_{(s(\sigma)-1)^\sigma} \wedge v_{s(\sigma)} < v_{(s(\sigma))^\sigma}$$

Comme $k^\sigma = k$ pour $k < s(\sigma)$ et $s(\sigma)^\sigma = t(\sigma)$, ceci peut être simplifier en $v_{s(\sigma)} < v_{t(\sigma)}$. □

Notons que nous n'avons utilisé qu'une propriété plus faible que l'injectivité. En efet, pour que le lemme soit vrai, il suffit que $v_{s(\theta)}$ et $v_{t(\theta)}$ soient différentes.

Notons également que si deux permutations σ et θ sont telles que $s(\sigma) = s(\theta)$ et $t(\sigma) = t(\theta)$, alors les contraintes éliminant ces symétries sont identiques. Il suffit donc d'ajouter une seule contrainte pour chaque paire i, j telle qu'il existe une permutation σ avec $i = s(\sigma)$ et $j = t(\sigma)$.

L'ensemble de ces paires peut être calculé avec l'algorithme de Schreier Sims [12]. Cet algorithme construit une suite de stabilisateurs $G_0, G_1, \ldots, G_n$ comme suit :

$$\begin{array}{rcl} G_0 & = & G \\ \forall i \in I^n, \ G_i & = & (i-1)_{G_{i-1}} \end{array}$$

Par définition,

$$\begin{array}{c} G_i = \{\sigma \in G : 0^\sigma = 0 \wedge \ldots \wedge (i-1)^\sigma = i-1\} \\ G_n \subseteq G_{n-1} \subseteq \ldots G_1 \subseteq G_0 \end{array}$$

L'algorithme de Schreier Sims calcule aussi les ensembles de représentant U_i qui sont les orbites de i dans G_i :

$$U_i = i^{G_i}$$

Par définition, U_i est l'ensemble des images de i par les permutations de G qui laissent $0, \ldots, (i-1)$ invariants.

Nous supposerons désormais que les groupes sont décrits par une suite de stabilisateurs et par les représentants associés.

Dans notre exemple, la suite de stabilisateurs est :

$$\begin{aligned} G_0 &= G \\ G_1 &= 0_{G_0} = \{[0,1,2,3,4,5],[0,2,1,3,5,4]\} \\ G_2 &= 1_{G_1} = \{[0,1,2,3,4,5]\} \end{aligned}$$

Tous les stabilisateurs suivants G_3, G_4, G_5 sont égaux à G_2.

Les représentants sont :

$$\begin{aligned} U_0 &= 0^{G_0} = \{0,1,2,3,4,5\} \\ U_1 &= 1^{G_1} = \{1,2\} \\ U_2 &= 2^{G_2} = \{2\} \\ U_3 &= 3^{G_3} = \{3\} \\ U_4 &= 4^{G_4} = \{4\} \end{aligned}$$

Théorème 2. *Etant donné un CSP injectif avec n variables $\mathcal{V}$, et étant donné des représentants U_i pour le groupe des symétries de variables du CSP, alors toutes les symétries de G peuvent être éliminées par au plus $n(n-1)/2$ contraintes binaires. Ces contraintes sont données par :*

$$\forall i \in I^n, \forall j \in U_i,\ i \neq j \rightarrow v_i < v_j \qquad (3)$$

Preuve. Par définition, pour chaque élément $j \in U_i$, il existe au moins une permutation $\sigma \in G_i$ telle que $i^\sigma = j$ et $j = t(\sigma)$. La réciproque est vraie : s'il existe une permutation σ telles que $i = s(\sigma)$ et telle $j = t(\sigma)$, alors $j \in U_i$. Donc, par le lemme 1, la contrainte (2) peut être réécrite en :

$$\forall i \in I^n, \forall j \in U_i,\ i \neq j \rightarrow v_i < v_j$$

Il y a au plus $\sum_{i=0}^{n-1}(|U_i|-1)$ contraintes. Toutes les permutations G_i laissent $0, \ldots, i-1$ invariants. Donc U_i est un sous ensemble de $\{i, \ldots, n-1\}$. Donc $|U_i| - 1 \leq n-i-1$, et le nombre de contraintes est borné supérieurement par $\sum_{i=0}^{n-1}(n-i-1) = n(n-1)/2$. □

Notons que d'après la première remarque du lemme 1, cette preuve n'utilise pas l'injectivité en général. Il suffit que les variables d'une même orbite soient distinctes. Le résultat reste vrai si le problème n'est pas injectif, mais si la condition suivant est vraie :

$$\forall i, j, k,\ j \in U_i \wedge k \in U_i \rightarrow v_j \neq v_k \qquad (4)$$

Dans notre exemple, les contraintes du théorème 2 sont :

$$v_0 < v_1,\ v_0 < v_2,\ v_0 < v_3,\ v_0 < v_4,\ v_0 < v_5,\ v_1 < v_2$$

Notons que certaines de ces contraintes sont redondantes. Par exemple, la contrainte $v_0 < v_2$ est impliquée par la première et la dernière contrainte, par transitivité. Nous pouvons donc la supprimer. Cette remarque est générale comme nous allons le voir dans la section suivante.

3.3 Un nombre linéaire de contraintes

Le résultat précédent peut être amélioré en utilisant la transitivité de la relation d'ordre. Etant donné $j \in I^n$, il se peut que j appartienne à plusieurs U_i. Dans ce cas, définissons $r(j)$le plus grand i différent de j tel que j appartienne à U_i. Si j n'appartient à aucun U_i autre que U_j, alors $r(j) = j$.

Avant d'énoncer notre résultat principal, prouvons ce qui suit.

Lemme 3. *Avec les notations qui précèdent, si $j \in U_i$ et $i \neq j$ alors $r(j) \in U_i$ et $r(j) < j$*

Preuve. Supposons que $j \in U_i$ et $i \neq j$. Par définition de U_i il existe une permutation $\sigma \in G_i$ telle que $i^\sigma = j$. Posons $k = r(j)$. Par définition de $r(j)$, $i \leq k$ et $j \in U_k$. Donc, il existe une permutation $\theta \in G_k$ telle que $k^\theta = j$. Posons $\nu = \sigma\theta^{-1}$. alors, $i^\nu = i^{\sigma\theta^{-1}} = j^{\theta^{-1}} = k$. De plus, $\nu \in G_i$ car $\sigma \in G_i$ et $\theta \in G_k \subseteq G_i$. Donc, $k \in U_i$. Le fait que $r(j) < j$ est une conséquence immédiate de la définition de $r(j)$.□

Nous pouvons maintenant énoncer notre résultat principal.

Théorème 4. *Avec les notations qui précèdent, étant donné un CSP injectif avec n variables $\mathcal{V}$, alors toutes les symétries de variables peuvent être éliminées avec au plus $n-1$ contraintes binaires. Ces contraintes sont données par :*

$$\forall j \in I^n,\ r(j) \neq j \rightarrow v_{r(j)} < v_j \qquad (5)$$

Preuve. Le nombre de contraintes (5) est au plus n par définition. Comme $r(0) = 0$ par définition, le nombre de contraintes est au plus $n-1$. Examinons

une des contraintes de (3) $c = (v_i < v_j)$, avec $j \in U_i$ et $i \neq j$. Nous cherchons à prouver que c est impliquée par les contraintes (5). Considérons la séquence $(j, r(j), r(r(j)), r(r(r(j))), \ldots)$. Supposons qu'elle ne rencontre jamais i. Nous avons $j \in U_i$ et $i \neq j$. D'après le lemme 3, $r(j) \in U_i$ et $r(j) < j$. Comme $r(j) \neq i$ par hypothèse, le lemme 3 peut être appliqué de nouveau. Par des applications répétées de ce lemme, nous construisons une séquence décroissante infinie incluse dans U_i. Ceci est impossible car U_i est fini. Donc il existe k tel que $i = r^k(j)$. De plus, nous avons établi que $r^k(j) \neq r^{k-1}(j), \ldots, r(r(j)) \neq r(j), r(j) \neq j$. Donc les contraintes $v_{r^k(j)} < v_{r^{k-1}(j)}, \ldots v_{r(r(j))} < v_{r(j)}, v_{r(j)} < v_j$ sont dans (5). Ensemble, elles impliquent $v_{r^k(j)} < v_j$ qui est la contrainte c. Nous avons donc prouvé que l'ensemble des contraintes de (3) est impliqué par les contraintes de (5). Comme (5) est un sous ensemble de (3), ces deux ensembles sont équivalents. Donc, par application du théorème 2, les contraintes (5) éliminent toutes les symétries. □

Notons que ce résultat reste valable si la condition (4) est vraie même si le problème n'est pas injectif :

Corollaire 5. *Avec les notations qui précèdent, étant donné un CSP avec n variables $\mathcal{V}$ tel que (4) est vrai, alors toutes les symétries de variables peuvent être éliminées avec au plus $n-1$ contraintes binaires. Ces contraintes sont données par (5).*

Dans notre exemple nous avons :

$$r(0) = 0, r(1) = 0, r(2) = 1, r(3) = 0, r(4) = 0, r(5) = 0$$

Donc les contraintes (5) du théorème 4 sont :

$$v_0 < v_1,\ v_0 < v_3,\ v_0 < v_4,\ v_0 < v_5,\ v_1 < v_2$$

Notons que la contrainte $v_0 < v_2$ n'apparaît plus.

Pour illustrer l'utilité du corollaire 5, prenons l'exemple des $n \times n$ reines. Il s'agit de colorier un échiquier de n cases de côté avec n couleurs de façon à ce qu'aucune ligne, colonne ou diagonale ne contienne deux fois la même couleur. Ceci peut être vu comme la recherche de n solutions disjointes au problème des n reines. Chaque solution est donnée par les cases d'une couleur donnée. Ce problème peut-être modélisé par une CSP avec n^2 variables, une par case de l'échiquier, et une contrainte tous différents par ligne, colonne et diagonale. Nous ordonnons les variables ligne par ligne (voir figure 2).

v_0	v_1	$\cdots$	v_{n-1}
v_n	v_{n+1}	$\cdots$	v_{2n-1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$v_{n(n-1)}$	$\cdots$	$\cdots$	v_{n^2-1}

Figure 2. Un échiquier.

Les symétries de variables proviennent des 8 symétries du carré. Calculons l'orbite de la première variable, v_0. Cette variable est dans un coin, et peut être envoyée sur n'importe quel autre coin. Donc nous avons :

$$U_0 = \{0, n-1, n^2-1, n(n-1)\}$$

Ensuite nous calculons le stabilisateur de v_0. La seule symétrie non triviale qui laisse le premier coin invariant est une réflexion sur la première diagonale. L'orbite de v_1 dans ce stabilisateur est donc :

$$U_1 = \{1, n\}$$

Les stabilisateurs suivants sont réduits à l'identité.

Les contraintes du problème sont telles que la condition (4) est satisfaite. En effet, deux coins donnés sont sur une même ligne, donc doivent prendre des valeurs différentes. De plus, les case de v_1 et v_n sont sur une même diagonale, donc leurs valeurs doivent être distinctes. Nous pouvons donc appliquer le corollaire 5 pour obtenir les contraintes suivantes :

$$v_0 < v_{n-1}, v_0 < v_{n(n-1)}, v_0 < v_{n^2-1}, v_1 < v_n$$

Ces contraintes éliminent toutes les symétries de variables pour ce problème.

4 Elimination des symétries de variables et de valeurs

Une méthode générale pour éliminer les symétries de valeurs est présentée dans [10]. Cette méthode utilise le groupe des symétries de valeurs du CSP. Nous allons prouver que cette méthode peut être utilisée en même temps que l'ajout de contraintes pour éliminer les symétries de variables.

Soit un CSP injectif $\mathcal{P}$ avec n variables v_i, et des domaines inclus dans I^k. Une méthode pour transformer toutes les symétries de valeurs en symétries de variables est présentée dans [2]. Un nouveau CSP $\mathcal{P}'$ est créé en ajoutant $n \times k$ variables binaires x_{ij} à $\mathcal{P}$. Ces variables sont reliées aux variables originelles par les contraintes suivantes :

$$\forall i \in I^n, j \in I^k,\ (x_{ij} = 1) \equiv (v_i = j)$$

La variable x_{ij} égale 1 si et seulement si la variable v_j égale j. L'ajout de ces variables et de ces contraintes ne change pas les solutions du CSP. De plus, les symétries de variables de $\mathcal{P}$ sont équivalentes aux permutations des lignes de la matrice x_{ij}, alors que les symétries de valeurs de $\mathcal{P}$ sont équivalentes aux permutations des colonnes de cette même matrice.

Soit X le vecteur de variables obtenu en concaténant les lignes de la matrice x_{ij}. Dans ce vecteur, les variables x_{ij} sont triées par ordre croissant de i puis de j.

Considérons une symétrie de valeur θ de $\mathcal{P}$. θ est une permutation des colonnes de la matrice. Cette symétrie est éliminée par la contrainte [1] :

$$X \preceq X^\theta \tag{6}$$

Soit X_i le vecteur des variables dans la ligne i de la matrice. La symétrie de valeurs θ envoie les variables d'une ligne de la matrice sur des variables de la même ligne :

$$\begin{aligned} X_i &= (x_{i0}, x_{i1}, \ldots, x_{i(k-1)}) \\ (X^\theta)_i &= (x_{i0^{\theta^{-1}}}, x_{i1^{\theta^{-1}}}, \ldots, x_{i(k-1)^{\theta^{-1}}}) \end{aligned}$$

Par définition de $\preceq$, nous avons que (6) est équivalent à la disjonction des contraintes suivantes :

$$\begin{gathered} X_0 \prec (X^\theta)_0 \\ X_0 = (X^\theta)_0 \wedge X_1 \prec (X^\theta)_1 \\ \vdots \\ X_0 = (X^\theta)_0 \wedge \ldots \wedge X_{i-1} = (X^\theta)_{i-1} \wedge X_i \prec (X^\theta)_i \\ \vdots \\ X_0 = (X^\theta)_0 \wedge \ldots \wedge X_{n-2} = (X^\theta)_{n-2} \wedge X_{n-1} \prec (X^\theta)_{n-1} \\ X_0 = (X^\theta)_0 \wedge \ldots \wedge X_{n-2} = (X^\theta)_{n-2} \wedge X_{n-1} = (X^\theta)_{n-1} \end{gathered}$$

Comparons X_i avec $(X^\theta)_i$. Soit a_i la valeur affectée à v_i. Alors $x_{i(a_i)} = 1$ et $x_{ij} = 0$ for $j \neq a_i$. De même, $x_{i(j)^{\theta^{-1}}} = 1$ si et seulement si $j^{\theta^{-1}} = a_i$, c'est-à-dire $a_i^\theta = j$. Donc, $X_i = (X^\theta)_i$ si et seulement si $a_i = (a_i)^\theta$, et $X_i \prec (X^\theta)_i$ si et seulement si $a_i < (a_i)^\theta$.

Donc (6) est équivalent à la disjonction des contraintes :

$$\begin{gathered} a_0 < (a_0)^\theta \\ a_0 = (a_0)^\theta \wedge a_1 < (a_1)^\theta \\ \vdots \\ a_0 = (a_0)^\theta \wedge \ldots \wedge a_{i-1} = (a_{i-1})^\theta \wedge a_i < (a_i)^\theta \\ \vdots \\ a_0 = (a_0)^\theta \wedge \ldots \wedge a_{n-2} = (a_{n-2})^\theta \wedge a_{n-1} < (a_{n-1})^\theta \\ a_0 = (a_0)^\theta \wedge \ldots \wedge a_{n-2} = (a_{n-2})^\theta \wedge a_{n-1} = (a_{n-1})^\theta \end{gathered}$$

Examinons l'un des termes de la disjonction, par exemple :

$$a_0 = (a_0)^\theta \wedge \ldots \wedge a_{i-1} = (a_{i-1})^\theta \wedge a_i < (a_i)^\theta$$

Ceci indique que θ laisse invariant $a_0, a_1, \ldots, a_{i-1}$. Dans ce cas a_i doit être le minimum parmi les valeurs sur lesquelles θ peut l'envoyer. Nous avons démontré le résultat suivant.

Lemme 6. *Avec les notations ci-dessus, a_i est minimal dans son orbite pour le groupe des symétries de valeurs qui laissent $a_0, a_1, \ldots a_{i-1}$ invariants.*

Ceci est équivalent à la méthode GE-tree [10] quand les variables et les valeurs sont testées en ordre croissant pendant la recherche.

D'après [1], il est correct de d'ajouter toutes les contraintes (1) pour $\mathcal{P}'$. En particulier, il est correct d'ajouter les contraintes (1) pour les symétries de variables de $\mathcal{P}$ avec toutes les contraintes (6). D'après le lemme 6, l'ensemble des contraintes (6) est équivalent à la méthode GE-tree pour éliminer les symétries de valeurs. Nous avons donc prouvé le résultat suivant.

Théorème 7. *Etant donné un CSP, son groupe de symétries de variables G_1, et son groupe de symétries de valeurs G_2, alors la combinaison de la méthode GE-tree avec les contraintes (1) calcule un ensemble de solutions $\mathcal{S}$ tel que :*

$$\forall S \in sol(\mathcal{P}), \exists \sigma \in G_1, \exists \theta \in G_2, \exists S' \in \mathcal{S},\ S^{\sigma\theta} = S'$$

Le corollaire du théorème 4 dit que l'ensemble des contraintes (1) est équivalent aux contraintes (5) pour les problèmes satisfaisant la condition (4). Ceci donne :

Corollaire 8. *Etant donné un CSP vérifiant (4), son groupe de symétries de variables G_1, et son groupe de symétries de valeurs G_2, alors la combinaison de la méthode GE-tree avec les contraintes (1) calcule un ensemble de solutions $\mathcal{S}$ tel que :*

$$\forall S \in sol(\mathcal{P}), \exists \sigma \in G_1, \exists \theta \in G_2, \exists S' \in \mathcal{S},\ S^{\sigma\theta} = S'$$

5 Résultats expérimentaux

Nous avons implanté un algorithme similaire à Nauty[5] pour calculer les automorphismes de graphes, ainsi qu'un algorithme de Schreier Sims [12]. Ces algorithmes ont été utilisés dans les exemples suivants. Dans notre implantation nous n'avons pas implémenté toute la méthode GE-tree, car elle requiert plus d'algorithmes de groups finis que ce que nous avons implémenté. Nous n'avons que codé le calcul des orbites dans le groups G_2 des symétries de valeurs. Ensuite, seule la valeur minimale dans chaque orbite est laissée dans le domaine de la variable v_0. Nous appelons notre méthode SBC (pour "Symmetry Breaking Constraints") dans ce qui suit, afin de la distinguer des méthodes précédemment publiées.

5.1 Graphes gracieux

Nous avons testé notre approche sur les graphes de [7] [8]. Les symétries de variables sont éliminées par les contraintes (5). Il y a une seule symétrie de valeurs non triviale, qui envoie a sur $e-a$, où e est le nombre d'arêtes du graphe. Les orbites pour les symétries de valeurs sont donc les ensembles $\{a, e-a\}$ pour $0 \leq a \leq e/2$. Donc on peut restreindre le domaine de $v_0$en gardant la plus petite valeur dans chacune de ces orbites.

Pour chaque graphe nous donnons le nombre de solutions du CSP (sol), la taille de l'arbre de recherche (noeud) et le temps de réponse. Les variables et les valeurs sont testées en ordre croissant. Nous donnons ces informations pour une recherche sans élimination de symétrie (no sym) et pour notre méthode (SBC). Dans ce cas le temps de réponse inclue le temps nécessaire pour la recherche des automorphismes du graphe et pour l'algorithme de Schreier Sims. Les temps de réponse sont mesurés sur un PC portable Dell Latitude D800 a 1.4 MHz sous Windows XP. Le CSP est résolu avec ILOG Solver 6.0[3].

graph	no sym			SBC		
	sol	noeud	temps	sol	noeud	temps
$K_3 \times P_2$	96	1518	0.12	8	83	0.01
$K_4 \times P_2$	1440	216781	13.6	30	1863	0.27
$K_5 \times P_2$	480	34931511	4454	2	53266	6.5
$K_6 \times P_2$				0	1326585	305

Table 1. Calcul de toutes les solutions pour les graphes gracieux.

Les tempes de réponse sont 30 fois plus petits que ceux obtenus par les méthodes GAP-SBDD et GAP-SBDS[8] sur un ordinateur environ 2 fois plus lent que le notre. La division par 15 du temps de réponse ne peut pas être entièrement imputée à la différence de logiciel de programmation par contraintes utilisé. Cet exemple montre que l'ajout de contraintes est beaucoup plus efficace que les méthodes modifiant la recherche de solution. Toutefois, notre méthode trouve deux fois plus de solutions. Voyons pourquoi sur le graphe $K_5 \times P_2$. Ce graphe a 10 sommets et 25 arêtes. Nous donnons la valeur des 10 variables du CSP correspondant pour les deux solutions trouvées :

$$(0, 4, 18, 19, 25, 23, 14, 6, 3, 1)$$

$$(0, 6, 7, 21, 25, 24, 22, 19, 11, 2)$$

L'application de la symétrie de valeurs non triviale à la deuxième solution donne :

$$(25, 19, 18, 4, 0, 1, 3, 6, 14, 23)$$

Appliquons la symétrie de variables suivante à cette solution :

$$[4, 3, 2, 1, 0, 9, 8, 7, 6, 5]$$

Cela nous donne la première solution !

Cet exemple montre que nous n'éliminons pas toutes les symétries qui sont la composition d'une symétrie de variables avec une symétrie de valeurs bien que chaque type de symétrie soit éliminé séparément.

5.2 Carrés magiques plus parfaits

Les carrés magiques plus parfaits ("most perfect magic squares"), étudiés dans [6], sont donnés comme exemple de CSP avec des symétries de valeurs complexes dans [10]. Les auteurs ont décidé d'utiliser une représentation duale pour que les symétries de variables du problème deviennent des symétries de valeurs, afin d'utiliser la méthode GE-tree. Il est prouvé dans [6] que les carrés magiques plus parfaits étaient en bijection avec les *carrés réversibles*. Un carré réversible est un carré de n cellules de côté ($n = 0$ modulo 4) rempli des nombres $1 \ldots n^2$ tel que (i) la somme de deux coins diagonalement opposés dans n'importe quel sous rectangle est égale à la somme des deux autres coins (ii) dans chaque ligne ou colonne, la somme du premier et dernier nombre égale la somme du deuxième et de l'avant dernier, etc (iii) la somme de deux nombres diamétralement opposés fait $n^2 + 1$.

Toute solution a $2^{n+1}((n/2)!)^2$ variantes symétriques [6]. Pour $n = 16$ ceci fait environ 2.13e+14.

Le modèle naturel pour ce problème a une variable par cellule du carré de domaine $\{1, 2, .., n^2\}$. Une contrainte tous différents est ajoutée en plus des 3 contraintes listées plus haut. Comme le problème est injectif, nous pouvons utiliser notre méthode pour éliminer les symétries de variable. Nous donnons pour différentes tailles de problème le temps nécessaire pour calculer les symétries du problème ainsi que le temps nécessaire pour calculer toutes les solutions non symétriques. Nous donnons également les temps de [10], obtenus avec GAP-SBDD et la méthode GE-tree sur un ordinateur deux fois plus lent que le notre. Une comparaison directe est difficile car le CSP utilisé n'est pas exactement le même. Toutefois, nous pouvons comparer le temps de calcul lié au groupe de symétries car il s'agit du même groupe dans les deux cas. Notre méthode requiert beaucoup moins de temps car le calcul n'a besoin d'être fait qu'une fois, avant de lancer la résolution du CSP.

Méthode	n	sols	sym	search
SBC	4	3	0.01	0.02
	8	10	0.09	0.39
	12	42	0.44	22.2
	16	35	4.6	275.6
GAP-SBDD	4	3	0.3	0.3
	8	10	5.4	125.4
	12	42	2745	12518
GE-tree	4	3	0.2	0.1
	8	10	0.7	90.0
	12	42	29.1	10901.8

Table 2. Calcul de toutes les solutions pour les carrés magiques plus parfaits.

6 Conclusion

Nous avons établi deux résultats (i) toutes les symétries de variables pouvaient être éliminées pas au plus $n-1$ contraintes binaires pour les CSP injectifs avec n variables (ii) la méthode GE-tree peut être utilisée en même temps que l'ajout de contraintes éliminant les symétries de variables.

Notre méthode peut être entièrement automatisée en utilisant des logiciels de calcul d'automorphisme de graphes tels que Nauty[5] et des algorithmes sur les groupes finis [12]. Nous avons implémenté ces algorithmes. Des expériences variées en montrent l'intérêt pratique. Toutefois, il est bon de noter que bien que notre méthode élimine toutes les symétries de variables et toutes les symétries de valeurs, elle n'enlève pas leurs combinaisons. Il nous semble judicieux de chercher à améliorer cet aspect.

Remerciements

Nous tenons à remercier Marie Puget et les relecteurs anonymes pour leurs remarques qui ont grandement contribué à améliorer la lisibilité de cet article.

Références

[1] Crawford, J., Ginsberg, M., Luks E.M., Roy, A. "Symmetry Breaking Predicates for Search Problems." In proceedings of KR'96, 148-159.

[2] P. Flener, A. M. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, T. Walsh. : "Breaking Row and Column Symmetries in Matrix Models. " Proceedings of CP'02, pages 462-476, 2002

[3] ILOG : ILOG Solver 6.0. User Manual. ILOG, S.A., Gentilly, France, Septembre 2003

[4] Lustig, I.J., and Puget, J.F. (2001). "Program Does Not Equal Program : Constraint Programming and its Relationship to Mathematical Programming," Interfaces 31(6), 29-53.

[5] Mc Kay, B. : "Practical Graph Isomorphism" Congr. Numer. 30, 45-87, 1981

[6] Dame Ollenrenshaw, K. "On most perfect or complete 8x8 pandiagonal magic squares" Proceedings Royal Society London, 407, 259-281, 1986.

[7] Petrie, K., Smith, B.M. : "Symmetry breaking in graceful graphs." In proceedings of CP'03, LNCS 2833, 930-934, Springer Verlag, 2003.

[8] Petrie, K. : "Combining SBDS and SBDD" Technical report APES-86-2004. Available from http ://www.dcs.st-and.ac.uk/~apes/apesreports.html"

[9] Puget, J.-F. : "On the Satisfiability of Symmetrical Constraint Satisfaction Problems." Proceedings of ISMIS'93 (1993), 350–361.

[10] Roney-Dougal C.M., Gent, I.P., Kelsey T., Linton S. : "Tractable symmetry breaking using restricted search trees" To appear in proceedings of ECAI'04.

[11] Roy. A., Luks, E. : "The complexity of symmetry-breaking formulas", *Annals of Mathematics and Artificial Intelligence* , 41 (2004), 19-45 (with A. Roy).

[12] Seress, A. : *Permutation Group Algorithms* Cambrige University Press, 2003.