Casser les symétries de variables dans un problème "presque" injectif

Philippe Vismara[1,2] **Rémi Coletta**[1]

[1] LIRMM, UMR5506 Université Montpellier II - CNRS, Montpellier, France
[2] MISTEA, UMR729 Montpellier SupAgro - INRA, Montpellier, France
{coletta,vismara}@lirmm.fr

Résumé

Une des techniques pour éliminer les symétries de variables est l'ajout de contraintes lexicographiques. Dans le cas général, le nombre de contraintes à ajouter au modèle pour éliminer toutes les symétries de variables est potentiellement exponentiel en n le nombre de variables [3]. Dans le cas particulier des problèmes injectifs (avec un AllDiff), le nombre de contraintes à ajouter est linéaire en le nombre de variables [8].

En se basant sur la contrainte globale de cardinalité [9], vue comme une généralisation de la contrainte AllDiff, nous caractérisons les problèmes "presque" injectifs par un paramètre μ correspondant au nombre de doublons.

Nous montrons ensuite que, pour ces problèmes "presque" injectifs, le nombre de contraintes à ajouter est majoré par $\binom{n}{\mu}$ et donc XP en termes de complexité paramétrée. Dans le cas où seules ν variables peuvent être affectées à un doublon, le nombre de contraintes est FPT en μ et ν.

Abstract

Lexicographic constraints are commonly used to break variable symmetries. In the general case, the number of constraint to be posted is potentially exponential in the number of variables. For injective problems (AllDiff), Puget's method[8] breaks all variable symmetries with a linear number of constraints.

In this paper we assess the number of constraints for "almost" injective problems. We propose to characterize them by a parameter μ based on Global Cardinality Constraint as a generalization of the AllDiff constraint. We show that for almost injective problems, variable symmetries can be broken with no more than $\binom{n}{\mu}$ constraints which is XP in the framework of parameterized complexity. When only ν variables can take duplicated values, the number of constraints is FPT in μ and ν.

1 Introduction

L'importance de la prise en compte des symétries dans la résolution d'un problème de satisfaction de contraintes est aujourd'hui largement reconnue. De nombreuses méthodes génériques on été développées pour éliminer ces symétries notamment de variables. Mais elles nécessitent généralement l'ajout d'un nombre de contraintes proportionnel au nombre de symétries qui peut s'avérer exponentiel. C'est le cas des contraintes lexicographiques [3] ou pour l'ajout dynamique de contraintes comme dans SBDS [5].

Mais sous certaines hypothèses, on peut se ramener à un nombre réduit de contraintes. Dans le cas des problèmes injectifs Pujet [8] a montré qu'il est possible d'éliminer les symétries de variables en posant un nombre linéaire de contraintes.

Entre ces deux extrêmes, cet article cherche à utiliser une approche de type complexité paramétrée pour évaluer le nombre de contraintes nécessaires à l'élimination des symétries de variables.

La notion de complexité paramétrée [4] permet de mesurer la complexité d'un problème NP-complet à l'aide d'une valeur k indépendante de la taille n des données. La complexité du problème est alors XP si elle s'exprime sous la forme $\mathcal{O}(n^k)$, voire FPT si elle est en $\mathcal{O}(f(k)n^c)$ où c est une constante et f une fonction quelconque, éventuellement exponentielle.

De nombreux problèmes de l'Intelligence Artificielle et plus particulièrement des CSP ont été analysés en termes de complexité paramétrée [6]. Par exemple, l'élimination des symétries de valeurs peut être réalisée avec une complexité paramétrée en $\mathcal{O}(2^k nd)$, où k est le nombre, potentiellement exponentiel, de symétries de valeurs [1].

De manière analogue, on peut bien sûr dire que le

nombre de contraintes nécessaires pour éliminer les symétries de variables est généralement FPT en le nombre de symétries.

L'objectif de cet article est d'essayer d'affiner ce critère, notamment dans le cas de problèmes "presque injectifs" .

2 Symétries dans les problèmes injectifs

2.1 symétries de variables

Une symétrie sur l'ensemble des variables $\{x_i\}_{i\in 1..n}$ est une permutation σ de $1..n$ telle que toute affectation $(x_i = v_i)_{i\in 1..n}$ est une solution si et seulement si $(x_{\sigma(i)} = v_i)_{i\in 1..n}$ en est une.

Une des démarches les plus courantes pour éliminer les symétries de variables consiste à poser des contraintes lexicographiques [3]. Étant donné un ordre $x_{i_1}, \dots x_{i_n}$ sur les variables, on pose, pour chaque symétrie de variable σ, une contrainte de la forme :

$$x_{i_1}, \dots, x_{i_n} \leq_{lex} x_{\sigma(i_1)}, \dots, x_{\sigma(i_n)} \qquad (1)$$

Malheureusement, le nombre de symétries (ou permutations) peut être exponentiel.

Généralement, un groupe $\mathcal{G}$ de symétries (ou permutations) est donné sous la forme d'un ensemble de générateurs S. On peut par exemple déterminer la taille et les générateurs du groupe de symétries (automorphismes) d'un graphe avec un logiciel comme Nauty [7] (ou encore Saucy). Ces logiciels sont très efficaces en pratique, comme l'écrit Brendan McKay : "*the theoretical worst-case efficiency of nauty is exponential. However, the worst cases are rather hard to find and for most classes of graphs nauty behaves as if it is polynomial.*"

Dans le cas de problèmes injectifs, on peut facilement montrer [8] que l'équation (1) se simplifie en $x_{i_k} \leq x_{\sigma(i_k)}$ où i_k est le plus petit indice tel que $\sigma(i_k) \neq i_k$. On aura donc moins de n^2 contraintes de différence pour l'ensemble des symétries.

Pour un indice i_k donné, les contraintes de la forme $x_{i_k} \leq x_{\sigma(i_k)}$ correspondent aux symétries σ qui sont des stabilisateurs de $i_1, .., i_{k-1}$ c.-à-d. au sous-groupe $\mathcal{G}^{[i_k]} = \{\sigma \in \mathcal{G} \mid \forall z < k,\ \sigma(i_z) = i_z\}$.

Plus précisément, on s'intéresse à toutes les images possibles de i_k par une symétrie qui laisse $i_1, .., i_{k-1}$ invariants. Cet ensemble d'images (ou *orbite*) peut se définir par $\Delta^{[i_k]} = \{\sigma(i_k) | \sigma \in \mathcal{G}^{[i_k]}\}$

Cette approche est intéressante parce qu'il est possible de déterminer tous les $\Delta^{[i_k]}$ sans énumérer $\mathcal{G}$, grâce à l'algorithme de Schreier-Sims.

2.2 L'algorithme de Schreier-Sims

On peut déjà simplifier le problème dans la mesure où il n'est pas nécessaire de considérer les n indices i_k mais seulement une partie de l'ensemble $1..n$.

En 1970, Sims a introduit la notion de *base* d'un groupe de permutations sur $1..n$. Une séquence $B = (\beta_1, \beta_2, ..., \beta_m)$, où $m \leq n$, est une base de $\mathcal{G}$ si aucune permutation de $\mathcal{G}$ n'est invariante pour B, à l'exception de l'identité. On parle de base parce que tout élément σ de $\mathcal{G}$ est parfaitement identifié par l'image $\sigma(B) = (\sigma(\beta_1), \sigma(\beta_2), ..., \sigma(\beta_m))$.

Une base B induit une chaîne de stabiliseurs $\mathcal{G} = \mathcal{G}^{[\beta_1]} \geq \mathcal{G}^{[\beta_2]} \geq \dots \geq \mathcal{G}^{[\beta_m]} \geq \{id.\}$ puisque chaque $\mathcal{G}^{[\beta_{k+1}]}$ (les stabilisateurs de $\beta_1, .., \beta_k$) est un sous-groupe de $\mathcal{G}^{[\beta_k]}$.

Si en plus ce sont des sous-groupes propres (non égaux), on dit que B est *non-redondante* et on montre que $2^{|B|} \leq |\mathcal{G}|$ c'est-à-dire $|B| \leq \log_2(|\mathcal{G}|)$

Étant donné un ensemble générateur S de $\mathcal{G}$, l'algorithme de Schreier-Sims construit incrémentalement une base B non-redondante et ajoute des générateurs à S afin que $S \cap \mathcal{G}^{[\beta_k]}$ soit un générateur de $\mathcal{G}^{[\beta_k]}$ (on parle alors de *strong generating set*).

Il détermine également l'orbite $\Delta^{[\beta_k]}$ correspondant à chaque β_k. Enfin, l'algorithme choisit un représentant (on parle de *transversal*) pour chaque valeur γ de $\Delta^{[\beta_k]}$, c'est-à-dire une permutation de $\mathcal{G}^{[\beta_k]}$, notée $u^{\gamma}_{\beta_k}$, qui associe β_k à γ (tout en laissant les $\beta_1, ..., \beta_{k-1}$ invariants).

Il est alors possible d'énumérer efficacement tous les éléments de $\mathcal{G}$, en utilisant une combinaison de représentants de chaque β_k. Si $U^{[k]} = \{u^{\gamma}_{\beta_k} | \gamma \in \Delta^{[\beta_k]}\}$ désigne l'ensemble de ces représentants pour β_k, on a $\mathcal{G} = \{\upsilon_{\beta_1} \circ \upsilon_{\beta_2} \circ \dots \circ \upsilon_{\beta_m} \mid \forall k \in 1..m, \upsilon_{\beta_k} \in U^{[k]}\}$

Il existe plusieurs variantes de l'algorithme de Schreier-Sims et une vaste littérature sur le sujet [2, 10]. La version déterministe la plus simple a une complexité en temps de $\mathcal{O}(n^2 \log^3 |\mathcal{G}| + |S| n^2 \log |\mathcal{G}|)$ et $\mathcal{O}(n^2 \log |\mathcal{G}| + |S| n)$ en mémoire.

En prenant pour base $B = (1, 2, ..., n)$ la variante proposé par Jerrum a une complexité $\mathcal{O}(n^5)$ en temps et $\mathcal{O}(n^2)$ en espace.

2.3 Un nombre polynomial de contraintes

Grâce à l'algorithme de Schreier-Sims on peut construire, en temps polynomial, les orbites $\Delta^{[\beta_k]}$ à partir d'un ensemble générateur S donné [1].

Chaque symétrie σ dans $\mathcal{G}$ appartient nécessairement à un sous-groupe $\mathcal{G}^{[\beta_k]}$. On peut donc remplacer la contrainte (1) par l'inégalité $x_{\beta_k} < x_{\sigma(\beta_k)}$ sachant

1. ou calculé avec *Nauty* en temps éventuellement exponentiel

que $\sigma(\beta_k) \in \Delta^{[\beta_k]}$. On en déduit un nombre polynomial d'inégalités :

$$\forall \beta_i \in B,\ \forall \gamma \in \Delta^{[\beta_i]}, \gamma \neq \beta_i,\ \text{on pose}\ \ x_{\beta_i} < x_\gamma \quad (2)$$

Le nombre d'inégalités est égal à $\sum_{\beta_i \in B}(|\Delta^{[\beta_i]}|-1)$ donc en $\mathcal{O}(n\, log_2|\mathcal{G}|)$ ou encore $\mathcal{O}(n^2)$

Dans [8], il est montré qu'on peut se limiter à une inégalité pour chaque γ, en ne considérant que $r(\gamma)$, le plus grand des β_i (différent de γ) tel que $\gamma \in \Delta^{[\beta_i]}$.

Formellement, soit $\mathcal{R}_\gamma = \{\beta_i \in B \setminus \{\gamma\} \mid \gamma \in \Delta^{[\beta_i]}\}$. Si $\mathcal{R}_\gamma \neq \emptyset$ on pose $r(\gamma) = \max(\mathcal{R}_\gamma)$ sinon $r(\gamma) = \gamma$.

On peut montrer [2] que les équations (2) sont équivalentes à un nombre linéaire de contraintes :

$$\forall \gamma \in 1..n \text{ tel que } r(\gamma) \neq \gamma,\ \text{on pose}\ \ x_{r(\gamma)} < x_\gamma \quad (3)$$

3 Gcc : relaxation du Alldiff aux problèmes "presque" injectifs

La `Gcc` (Global Cardinality Constraint) a été introduite dans [9]. C'est une contrainte largement utilisée dans la modélisation de problèmes industriels et elle est disponible dans la plupart des résolveurs de contraintes existants. Elle exprime une propriété très générique : elle contraint le nombre d'occurrences de chaque valeur présente dans les domaines. En d'autres termes, cette contrainte traite globalement une conjonction de `AtLeast` (imposant qu'une valeur v apparaisse au moins un certain nombre de fois) et `AtMost` (imposant qu'une valeur v apparaisse au plus un certain nombre de fois).

Définition 1 *Une contrainte* `Gcc`*, notée* $Gcc(X, lb, up)$ *porte sur un ensemble de variables* X *et de 2 fonctions* lb *et* ub *de* $\bigcup_{x\in X} D(x)$ *dans* $\mathcal{N}$*. La contrainte* `Gcc` *est vérifiée si pour chaque valeur* $v \in \bigcup_{x\in X} D(x)$ *le nombre de variables de* X *assignées à* v *est compris entre* $lb(v)$ *et* $ub(v)$*.*

Soit $N =< X, D, C >$ un réseau de contraintes avec $Gcc(X, lb, ub) \in C$. On pose $\mu = 1 + \sum_{v\in D}(ub(v) - 1)$

Si $\mu = 1$, alors la Gcc est un Alldiff, on a un problème injectif. Si μ est "petit", alors on a un problème "presque" injectif. Le paramètre μ permet donc de mesurer l'écart, en nombre de doublons autorisés, par rapport à un problème parfaitement injectif.

4 Généralisation aux problèmes "presque" injectifs

Considérons un problème presque injectif admettant au plus μ doublons.

Pour chaque symétrie de variables $\sigma \in \mathcal{G}^{[\beta_k]}$, la contrainte lexicographique (1) n'est plus équivalente à $x_{\beta_k} < x_{\sigma(\beta_k)}$ puisque $x_{\beta_k} = x_{\sigma(\beta_k)}$ devient possible. En cas d'égalité, tester la contrainte lexicographique revient à déterminer le prochain $\beta_z > \beta_k$ tel que $x_{\beta_z} \neq x_{\sigma(\beta_z)}$, avec nécessairement $\sigma(\beta_z) \neq \beta_z$.

On peut supposer ici que $|B| = n$. Sinon il suffit d'ajouter à la fin de la base les valeurs manquantes dans un ordre quelconque. Les générateurs calculés par l'algorithme de Schreier-Sims restent valables pour cette base étendue.

Formellement, étant donnée une symétrie de variables $\sigma \in \mathcal{G}^{[\beta_k]}$, considérons la suite croissante $i^1_\sigma, i^2_\sigma, \ldots i^t_\sigma$ des indices de B pour lesquels $\sigma(i) \neq i$. Puisque $\sigma \in \mathcal{G}^{[\beta_k]}$ on a $i^1_\sigma = \beta_k$.

Soit ρ le plus petit indice de $1..t$, s'il existe, tel que :

$$|\{i^z_\sigma\}_{z\in 1..\rho} \cup \{\sigma(i^z_\sigma)\}_{z\in 1..\rho}| > \mu$$

Si t est trop petit pour permettre de couvrir les μ variables, on pose $\rho = t \leq \mu$.

Sinon, dans le cas général : $\frac{\mu}{2} < \rho \leq \mu + 1$.

Pour un problème injectif, la contrainte lexico (1) était remplacée par la contrainte $x_{i^1_\sigma} < x_{\sigma(i^1_\sigma)}$

Dans le cas où μ valeurs identiques sont possibles, la contrainte lexico (1) devient :

$$x_{i^1_\sigma}, x_{i^2_\sigma}, \ldots, x_{i^\rho_\sigma} \leq_{lex} x_{\sigma(i^1_\sigma)}, x_{\sigma(i^2_\sigma)}, \ldots, x_{\sigma(i^\rho_\sigma)} \quad (4)$$

Au final, on aura donc au plus $\binom{n}{\mu+1}$ contraintes (4) pour l'ensemble des symétries.

On obtient ainsi un nombre de contraintes lexicographiques qui n'est plus potentiellement exponentiel mais XP en fonction de μ puisque $\binom{n}{\mu} < n^\mu$

Pour construire l'ensemble des contraintes (4), il faut énumérer l'ensemble $\mathcal{G}$ des symétries ce qui peut être réalisé en $\mathcal{O}(n|\mathcal{G}|)$, grâce aux ensembles $U^{[k]}$ calculés par l'algorithme de Schreier-Sims.

On peut remarquer que chaque contrainte (4) se décompose en contraintes de la forme :

$$x_{i^1_\sigma} \leq x_{\sigma(i^1_\sigma)} \quad (5a)$$

$$x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)} \rightarrow x_{i^2_\sigma} \leq x_{\sigma(i^2_\sigma)} \quad (5b)$$

$$\ldots$$

$$x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)} \wedge \ldots \wedge x_{i^{\rho-1}_\sigma} = x_{\sigma(i^{\rho-1}_\sigma)} \rightarrow x_{i^\rho_\sigma} \leq x_{\sigma(i^\rho_\sigma)} \quad (5c)$$

Il y aura $\sum_{\beta_i \in B}(|\Delta^{[\beta_i]}|-1)$ contraintes de type (5a), au plus $\binom{n}{4}$ contraintes de type (5b) et finalement au plus $\binom{n}{\mu+2}$ contraintes de type (5c).

C'est une majoration très grossière et dans le pire des cas. En pratique on peut éliminer des contraintes. Par exemple, si on a posé une contrainte (5a) de

2. Chaque $x_{\beta_i} < x_\gamma$ est équivalent à la série d'inégalités $x_{\beta_i = r(\ldots r(r(\gamma)))} < \ldots < x_{r(r(\gamma))} < x_{r(\gamma)} < x_\gamma$

la forme $x_a \leq x_b$ il est inutile de poster toutes les contraintes finissant par « $\rightarrow x_a \leq x_b$ ».

Par ailleurs, les contraintes (5a) peuvent être remplacées par un nombre linéaire de contraintes de la forme $x_{r(j)} \leq x_j$.

5 Discussion

Nous avons établi que dans le cas d'un problème presque injectif, le nombre de contraintes lexicographiques nécessaires pour éliminer toutes les symétries de variables est en $\mathcal{O}(\binom{n}{\mu})$.

Ce nombre dépend des symétries mais aussi de la base B utilisée. Supposons que δ soit le plus petit indice (dans l'ordre de B) d'une variable x_δ dont la valeur n'est pas unique. Pour chaque symétrie $\sigma \in \mathcal{G}^{[\beta_k]}$ telle que $i^1_\sigma = \beta_k < \delta$, tout se passe comme si le problème était injectif. Dans le cas extrême où $\delta > |B|$ (avec $|B| < n$), on se ramène au cas injectif.

On pourrait donc envisager d'adapter la base B pour que les indices des variables ayant une valeur répétée soient placés à la fin de B.

Une adaptation dynamique de la base serait probablement très coûteuse. D'une part parce que les contraintes de type (2) ne pourraient être activées dynamiquement que si on est certain d'avoir $\beta_k < \delta$. Par ailleurs, il est possible de réordonner B pour déplacer les indices correspondant à des doublons vers la fin. Mais la simple permutation de deux indices contigus coûte $\mathcal{O}(n^4)$[2].

De façon statique, on peut imaginer des cas particuliers où les doublons ne peuvent concerner qu'un sous ensemble de variables. Si on peut construire une base B avec les indices des autres variables, on peut éliminer les symétries comme dans le cas injectif.

Supposons enfin qu'il n'y ait que ν variables pouvant être affectées à un doublon. Dans ce cas, la base B peut être construite de telle sorte que les indices de ces ν variables soient placés à la fin. On est alors ramené au cas injectif pour toute symétrie σ telle que $i^1_\sigma < \beta_{n-\nu}$ et l'ensemble des symétries de $\mathcal{G} \setminus \mathcal{G}^{[\beta_{n-\nu}]}$ se ramène à un nombre linéaire de contraintes. Seules les symétries de $\mathcal{G}^{[\beta_{n-\nu}]}$ nécessiteront une contrainte lexico (4). Puisque ces contraintes ne concernent que ν variables, leur nombre est majoré par $\binom{\nu}{\mu}$. Le nombre total de contrainte est donc dans ce cas en $\mathcal{O}(\binom{\nu}{\mu} + n)$ c'est-à-dire FPT en ν et μ.

Influence de domaines hétérogènes

Dans la section 4, nous avons donné une borne théorique du nombre de contraintes à poser. La majoration ne tient pas compte des domaines initiaux des variables, dans le cas où ils ne sont pas tous égaux.

En effet, si $D(x_{i^1_\sigma}) \cap D(x_{\sigma(i^1_\sigma)}) = \emptyset$, il est inutile de poser la contrainte 5b et toutes les contraintes qui contiennent $x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)}$ en partie gauche.

La combinaison de la contrainte de cardinalité et de la répartition initiale des domaines peut aussi interdire des combinaisons d'égalité : Par exemple si $D(x_1) \cap D(x_2) = \{v\} = D(x_3) \cap D(x_4)$ et $ub(v) = 3$, alors la `Gcc` interdit de fait d'avoir $x_1 = x_2$ et $x_3 = x_4$ simultanément. Donc toutes les contraintes 5c avec $x_1 = x_2 \wedge x_3 = x_4$ en partie gauche n'ont pas besoin d'être posées.

Plus généralement, lors de la génération des contraintes d'un réseau $N =< X, D, C >$. Avant de poser la contrainte $x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)} \wedge \ldots \wedge x_{i^{\rho-1}_\sigma} = x_{\sigma(i^{\rho-1}_\sigma)} \rightarrow x_{i^\rho_\sigma} \leq x_{\sigma(i^\rho_\sigma)}$, nous proposons de tenter de résoudre le problème $N' =< X, D, C' >$ avec $C' = \{x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)}, \ldots, x_{i^{\rho-1}_\sigma} = x_{\sigma(i^{\rho-1}_\sigma)}\} \cup \{Gcc\}$, le sous problème constitué la `Gcc` et des égalités de la gauche de cette contrainte. Si N' est insatisfiable, alors on ne pose pas la contrainte.

Ce test est polynomial, car la classe des réseaux de contraintes composés d'une contrainte `Gcc` et de contraintes d'égalités est polynomiale pour le problème de satisfaction. L'idée du codage consiste en une modification du problème de flot utilisé pour résoudre la contrainte de `Gcc` [9]. Pour toute paire de variables (x_i, x_j) telle que $x_i = x_j$, on supprime les nœuds x_i et x_j, ainsi que leur arêtes et on ajoute un noeud $x_{(i,j)}$ et $\forall v \in D(x_i) \cap D(x_j)$, on ajoute une arrête $(x_{(i,j)}, v)$ avec $(0, 2)$ comme flots minimum et maximum.

Cette démarche fournit ainsi un moyen supplémentaire de réduire le nombre de contraintes pour les problèmes presque injectifs, en complément des résultats que nous avons établis en complexité paramétrée.

6 Conclusion

Nous avons montré qu'il est possible d'éliminer les symétries de n variables d'un problème presque injectif en posant moins de $\binom{n}{\mu}$ contraintes lexicographiques, où μ correspond au nombre de doublons. On obtient bien un nombre XP de contraintes lexicographiques.

Dans le cas où il n'y a que ν variables qui peuvent être affectées à un doublon, le nombre des contraintes posées est en $\mathcal{O}(\binom{\nu}{\mu} + n)$ donc FPT.

Lorsque les domaines des variables sont hétérogènes, nous avons montré qu'un test polynomial (basé sur la `Gcc`) permet d'écarter des contraintes lexicographiques inutiles.

Références

[1] C. Bessiere, E. Hebrard, B. Hnich, Z. Kiziltan, C.-G. Quimper, and T. Walsh. The parameterized

complexity of global constraints. In *Proceedings AAAI*, pages 235–240, 2008.

[2] Gregory Butler. *Fundamental Algorithms for Permutation Groups*, volume 559 of *Lecture Notes in Computer Science*. Springer, 1991.

[3] James Crawford, Matthew Ginsberg, Eugene Luks, and Amitabha Roy. Symmetry-breaking predicates for search problems. In *KR*, pages 148–159. Morgan Kaufmann, 1996.

[4] Rodney G. Downey, Michael R. Fellows, and Ulrike Stege. Parameterized complexity : A framework for systematically confronting computational intractability. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, volume 49, pages 49–99, 1997.

[5] Ian P. Gent, Warwick Harvey, and Tom Kelsey. Groups and constraints : Symmetry breaking during search. In *CP 2002*, volume 2470 of *Lecture Notes in Computer Science*, pages 415–430, 2002.

[6] Georg Gottlob and Stefan Szeider. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. *Comput. J.*, 51(3) :303–325, 2008.

[7] B. D. McKay. *nauty user's guide (version 2.4)*. Technical report, Australian National University, Computer Science Department, `http://cs.anu.edu.au/~bdm/nauty/`, 2009.

[8] Jean-François Puget. Breaking symmetries in all different problems. In *Proceedings IJCAI'05*, pages 272–277, 2005.

[9] J-C. Régin. Generalized arc consistency for global cardinality constraint. *AAAI*, pages 209–215, 1996.

[10] A. Seress and Á. Seress. *Permutation group algorithms*. Cambridge tracts in mathematics. Cambridge University Press, 2003.