Breaking variable symmetry in almost injective problems

Philippe Vismara[1,2] and Remi Coletta[1]

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

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[11] 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

The importance of symmetry is now widely recognized in Constraint Satisfaction Problems. Many methods have been devised to break symmetry especially for variables. Most of them require to post a number of constraints which is equal to the number of symmetries. This is the case of lexicographic constraints [6] or dynamic constraints in SBDS [8].

Unfortunately, the number of variable symmetries can be exponential in the general case. Even so, Puget [11] has shown that, for injective problems, symmetry can be broken with a linear number of constraints.

Between these two extreme, the aim of this paper is to use a parameterized complexity approach in order to evaluate the number of constraints required to break variable symmetries.

Parameterized complexity [7] offers a measure of complexity for NP-complete problems which is based on an isolate parameter k independent from the size n. The complexity of a problem is XP if it is in the form $\mathcal{O}(n^k)$ or FPT if it is in $\mathcal{O}(f(k)n^c)$ where c is a constant and f any function, even exponential.

Many problems in Artificial Intelligence, especially CSP, have been analyzed with parameterized complexity [9]. For instance, breaking value symmetries has a fixed-parameter complexity in $\mathcal{O}(2^k nd)$, where k is the potentially exponential number of value symmetries [2].

By analogy, we can claim that variable symmetry can be broken with number of lexicographic constraints which is FTP in the number of symmetries. In this paper we try to propose new parameters to measure the number of required constraints.

2 Breaking Variable Symmetry

A variable symmetry is a permutation σ on the set of variables $\{x_i\}_{i\in 1..n}$ that maps solutions onto solutions. More formally, any assignment $(x_i = v_i)_{i\in 1..n}$ is a solution if and only if $(x_{\sigma(i)} = v_i)_{i\in 1..n}$ is also a solution.

The general method for breaking variable symmetry is to add lexicographical constraints [6]. Given an order $x_{i_1}, \ldots x_{i_n}$ on variables, we post, for any symmetry σ, the lexicographical ordering constraint:

$$x_{i_1}, \ldots, x_{i_n} \leq_{lex} x_{\sigma(i_1)}, \ldots, x_{\sigma(i_n)} \tag{1}$$

Unfortunately, the number of variable symmetries can be exponential. It is equal to the size of the corresponding permutation group $\mathcal{G}$. This group is generally given as a set S of generators. For instance, Nauty [10] or Saucy are well known programs to compute the permutation group of a graph (automorphisms). Even if their theoretical worst-case time complexity is exponential, they are very efficient in practice.

For injective problems, Puget[11] has shown that equation (1) can be simplified into $x_{i_k} \leq x_{\sigma(i_k)}$, where i_k is the smallest index such that $\sigma(i_k) \neq i_k$. Hence there are less than n^2 such constraints for all variable symmetries.

Given a variable index i_k, any symmetry σ for which we post the constraint $x_{i_k} \leq x_{\sigma(i_k)}$ is a stabilizer of $i_1, .., i_{k-1}$. So σ belongs to the stabilizer subgroup $\mathcal{G}^{[i_k]} = \{\sigma \in \mathcal{G} \mid \forall z < k,\ \sigma(i_z) = i_z\}$. More precisely, we need to compute the image (orbit) of i_k by any symmetry that leaves $i_1, .., i_{k-1}$ invariant. This set is defined by $\Delta^{[i_k]} = \{\sigma(i_k) | \sigma \in \mathcal{G}^{[i_k]}\}$

This approach is interesting because it is possible to compute all sets $\Delta^{[i_k]}$ without enumerating $\mathcal{G}$, thanks to Schreier-Sims' algorithm.

2.1 Schreier-Sims' algorithm

In 1970, Sims introduced the notion of *base* for a permutation group. A sequence $B = (\beta_1, \beta_2, ..., \beta_m)$, where $m \leq n$, is a base for $\mathcal{G}$ if the only permutation which fixes each of the points in B is the identity.

The word "base" is used because an element σ of the group $\mathcal{G}$ is uniquely determined by the image $\sigma(B) = (\sigma(\beta_1), \sigma(\beta_2), ..., \sigma(\beta_m))$.

A base B induces a stabilizer chain $\mathcal{G} = \mathcal{G}^{[\beta_1]} \geq \mathcal{G}^{[\beta_2]} \geq \ldots \geq \mathcal{G}^{[\beta_m]} \geq \{id.\}$ since $\mathcal{G}^{[\beta_{k+1}]}$ (the permutations that fix $\beta_1, .., \beta_k$) is a subgroup of $\mathcal{G}^{[\beta_k]}$.

If every $\mathcal{G}^{[\beta_{k+1}]}$ is a proper (not included) subgroup of $\mathcal{G}^{[\beta_k]}$, the base B is called *nonredundant* and $2^{|B|} \leq |\mathcal{G}|$ thus $|B| \leq \log_2(|\mathcal{G}|)$.

Given any set S of generators of the group $\mathcal{G}$, the Schreier-Sims's algorithm computes incrementally a nonredundant base B. This approach is analogous

to Gaussian elimination in Algebra. The algorithm adds new generators to S such that $S \cap \mathcal{G}^{[\beta_k]}$ is a generator of $\mathcal{G}^{[\beta_k]}$. The resulting base is called a *strong generating set*.

For each β_k in B, the algorithm computes the orbit $\Delta^{[\beta_k]}$ and chooses, for each value γ in $\Delta^{[\beta_k]}$, one representative permutation $u^{\gamma}_{\beta_k} \in \mathcal{G}^{[\beta_k]}$ such that $u^{\gamma}_{\beta_k}(\beta_k) = \gamma$. Thanks to this set of representatives, one can easily enumerate $\mathcal{G}$.

Let $U^{[k]} = \{u^{\gamma}_{\beta_k} | \gamma \in \Delta^{[\beta_k]}\}$ be the set of representatives for β_k.

Then $\mathcal{G} = \{\upsilon_{\beta_1} \circ \upsilon_{\beta_2} \circ \ldots \circ \upsilon_{\beta_m} \mid \forall k \in 1..m, \upsilon_{\beta_k} \in U^{[k]}\}$.

There exists several variants of Schreier-Sims' algorithm and a vast literature on this topic[5, 13]. The simplest deterministic version has a time complexity in $\mathcal{O}(n^2 \log^3 |\mathcal{G}| + |S| n^2 \log |\mathcal{G}|)$ and $\mathcal{O}(n^2 \log |\mathcal{G}| + |S| n)$ in space.

If we choose $B = (1, 2, ..., n)$, the Jerrum's variant has a time complexity in $\mathcal{O}(n^5)$ and $\mathcal{O}(n^2)$ in space.

2.2 A polynomial number of constraints

With Schreier-Sims' algorithm one can enumerate in polynomial time all orbits $\Delta^{[\beta_k]}$ from a given[3] generating set of $\mathcal{G}$.

Since any symmetry σ must belong to a subgroup $\mathcal{G}^{[\beta_k]}$, each constraint (1) is equivalent to an inequality $x_{\beta_k} < x_{\sigma(\beta_k)}$ with $\sigma(\beta_k) \in \Delta^{[\beta_k]}$. Hence all variable symmetries can be broken with the following constraints:

$$\forall \beta_i \in B,\ \forall \gamma \in \Delta^{[\beta_i]}, \gamma \neq \beta_i,\ \text{we post}\ \ x_{\beta_i} < x_{\gamma} \tag{2}$$

The total number of inequalities is $\sum_{\beta_i \in B}(|\Delta^{[\beta_i]}| - 1)$ so it is in $\mathcal{O}(n\, log_2 |\mathcal{G}|)$ or in $\mathcal{O}(n^2)$.

In [11], Puget has shown that equation (2) can be reduced to one inequality for each γ in $\Delta^{[\beta_i]}$. The principle is to associate each γ to the larger β_i (different from γ) such that $\gamma \in \Delta^{[\beta_i]}$.

Formally, let $\mathcal{R}_{\gamma} = \{\beta_i \in B \setminus \{\gamma\} \mid \gamma \in \Delta^{[\beta_i]}\}$. If $\mathcal{R}_{\gamma} \neq \emptyset$ let us define $r(\gamma) = \max(\mathcal{R}_{\gamma})$ otherwise $r(\gamma) = \gamma$.

One can prove[4] that equation (2) is equivalent to the following linear number of constraints:

$$\forall \gamma \in 1..n \text{ such that } r(\gamma) \neq \gamma,\ \text{we post}\ \ x_{r(\gamma)} < x_{\gamma} \tag{3}$$

3 Gcc : a relaxation of Alldiff for almost injective problems

A constraint network that involves an Alldiff constraint is injective and we showed in Section 2 that the number of constraints required to break all variables symmetries is polynomial. The Alldiff imposes that each value be taken at

[3] or computed with *Nauty* potentially in exponential time

[4] by transitivity, $x_{\beta_i} < x_{\gamma}$ is derived from $x_{\beta_i = r(\ldots r(r(\gamma)))} < \ldots < x_{r(r(\gamma))} < x_{r(\gamma)} < x_{\gamma}$

most 1 time, which is quite restrictive in terms of modelisation. In many practical applications, we may want to impose that a value should appear at least l and/or at most u times. With this purpose, the Global Cardinality Constraint `Gcc` has been introduced in [12] to deal globally with a conjunction of `AtLeast` (stating that a value has to appear at least a given number of times) and `AtMost` (stating that a value has to appear at most a given number of times). The `Gcc` is widely used to model industrial problems and is available in almost all existing constraints solvers.

Definition 1 (Global Cardinality Constraint). *A `Gcc` constraint, denoted* $Gcc(X, lb, up)$ *involves a set of variables* X *and two functions* $lb, ub : \bigcup_{x \in X} D(x) \rightarrow \mathcal{N}$. *The `Gcc` constraint is satisfied if for each value* $v \in \bigcup_{x \in X} D(x)$ *the number of variables in* X *assigned to* v *is between* $lb(v)$ *and* $ub(v)$.

This constraint does not guarantee the problem to be injective, but in the remainder of this section we show that a problem with `Gcc` constraint may be *almost injective*. To measure the distance of a given problem N to a perfectly injective problem, we introduce the parameter $\mu(N)$, maximum number of variables that can be equal simultaneously. If $\mu(N) = 0$, then the problem N is perfectly injective. If $\mu(N)$ is *small* we call the problem N *almost injective*.

We provide below a first upper bound of $\mu(N)$ for a constraint network N which contains a `Gcc` constraint, by considering in a static way the different upper bounds of the values.

Property 1. Let $N =< X, D, C >$ be a constraint network, with $Gcc(X, lb, ub) \in C$. $\mu \leq \sum_{v \in D \ s.t. \ ub(v)>1} ub(v)$.

This bound does not takes into account neither the overlap between variables domains nor the lower bound of values. To deal with both the lower bounds and the variables domains and then achieve a better upper bound of μ, we have to look at the implementation of the `Gcc`.

The `Gcc` constraint propagation algorithm proposed in [12] consists in finding a flow in a bipartite graph, such that one of Figure 1. The set of the set of nodes is the union of the variables involved in the `Gcc` and their values and the two special nodes s and t. The flow between the source s and a variable x is exactly 1, stating that each variable has to be assigned to a single value. The flow between a variable x and a value v in its domain is either 0 or 1, depending if x is assigned to v or not. The flow between a value v and s is constrained by the lower and upper bounds of the value in the `Gcc`.

Property 2. Given a constraint network $N =< X, D, C >$ with $Gcc(X, lb, ub) \in C$, $\mu(N)$ is bounded by the number of variables minus the minimal number of values which can be assigned to exactly one variable while respecting the `Gcc`.

Computing this minimal number of values is more complex than establishing the upper bound of Prop. 1. We propose to compute it using the CSP encoding described below. As described above the underlying algorithm of the `Gcc` relies on a flow problem. But, the flow problem itself has been encapsulated in a constraint

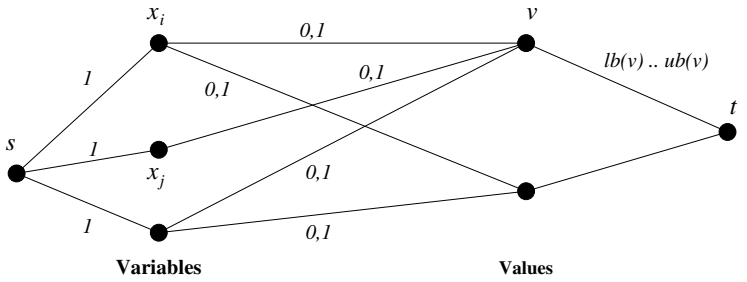

Fig. 1. The network used for GCC propagation

in [4], allowing to solve problems involving flow and additional constraints, for instance large workforce scheduling problems [1] . In this flow constraint, the flow allowed to an edge (i, j) is expressed as a CSP variable $x_{flow(i,j)}$.

To compute the minimal number of values which can be assigned to exactly one variable while respecting the `Gcc`, we propose to reuse the encoding of the `Gcc` enriched with extra variables $nb_occ(v_i)$to count the number of occurrences of a value. $nb_occ(v_i)$ is the variable expressing the flow between v_i and t. Minimize $\sum_{v_i \in D}(nb\_occ(v_i) = 1)$ is equivalent to compute the minimal number of values which can be assigned to exactly one variable while respecting the `Gcc`.

Lemma 1. *The problem of determining if* $\sum_{v_i \in D}(nb\_occ(v_i) = 1) \leq N$ *is NP-hard.*

Proof (sketch of): Consider a $Gcc(X, lb, up)$ with $lb(v) = 0, ub(v) = |X|$, finding $\sum_{v_i \in D}(nb\_occ(v_i) \geq 1) \leq N$ exactly fits the definition of the $atmostNvalues(X, N)$ constraint, which was shown NP-hard in [3] by a reduction to $3 - SAT$. For $\sum_{v_i \in D}(nb\_occ(v_i) \geq 1) \leq N$ the proof is identical except the point 3-SAT is replaced by exactly-1 3SAT (also called 1-in-3 SAT).

4 Generalization to "almost injective" problems

Suppose that an almost injective problem admits no more than μ simultaneous pairs of equal variables.

For each variable symmetry $\sigma \in \mathcal{G}^{[\beta_k]}$, constraint (1) is no longer equivalent to $x_{\beta_k} < x_{\sigma(\beta_k)}$ because x_{β_k} can be equal to $x_{\sigma(\beta_k)}$. In such a case, checking the lexicographical constraint involves to find the next $\beta_z > \beta_k$ such that $x_{\beta_z} \neq x_{\sigma(\beta_z)}$ and so $\sigma(\beta_z) \neq \beta_z$.

Given any symmetry $\sigma \in \mathcal{G}^{[\beta_k]}$, consider the increasing sequence $i^1_\sigma, i^2_\sigma, \ldots i^t_\sigma$ of the elements[5] of B for which $\sigma(i) \neq i$. Since $\sigma \in \mathcal{G}^{[\beta_k]}$ we have $i^1_\sigma = \beta_k$.

For injective problems we have seen that lexicographic constraint (1) can be replaced by the constraint $x_{i^1_\sigma} < x_{\sigma(i^1_\sigma)}$. For almost injective problem with no more than μ simultaneous pairs of equal variables, constraint (1) simplifies in:

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

[5] If $|B| < n$ we can add missing values at the end of the base. It does not affect the strong generating set computed with Schreier-Sims' algorithm.

where $\rho = \min(\mu + 1, t)$.

Because there are no more than $\binom{n}{2\mu}$ constraints (4) for all symmetries in $\mathcal{G}$ and $\binom{n}{k} < n^k$, we have the following lemma:

Lemma 2. *Given a CSP where no more than μ variables can be equal simultaneously, all variable symmetries can be broken with a number of constraints which is XP in μ.*

To compute constraints (4), we have to enumerate the whole group $\mathcal{G}$. This can be done in $\mathcal{O}(n|\mathcal{G}|)$ thanks to sets $U^{[k]}$ computed by Schreier-Sims' algorithm.

By definition, constraint (4) is equivalent to the following constraints:

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

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

$$\dots$$

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

There are $\sum_{\beta_i \in B}(|\Delta^{[\beta_i]}| - 1)$ constraints of type (5a), no more than $\binom{n}{4}$ constraints (5b) and finally no more than $\binom{n}{\mu+2}$ constraints (5c). It is a crude upper bound in the worst case. In practice some constraints can be discarded. For instance, if we post a constraint (5a) like $x_a \leq x_b$, it is unnecessary to post any constraint that ends with "$\rightarrow x_a \leq x_b$". Moreover, constraints (5a) are equivalent to a linear number of constraints of the form $x_{r(j)} \leq x_j$.

The total number of constraints required to break symmetry also depends on the ordering of the variables in base B. For instance, suppose that δ is the first index in B such that x_δ takes a duplicate value. For each $\sigma \in \mathcal{G}^{[\beta_k]}$ such that $i^1_\sigma = \beta_k < \delta$, the lexicographic constraint is equivalent to $x_{i^1_\sigma} < x_{\sigma(i^1_\sigma)}$. All these inequalities can be reduced to a linear number of constraints of the form $x_{r(j)} \leq x_j$. The special case when $\delta > |B|$ (with $|B| < n$) is completely equivalent to injective problems.

Therefore, we can try to rearrange base B in order to set, at the end of the list, the indexes of the variables that take duplicate values.

A dynamic rearrangement could be expensive because permuting a single pair of contiguous indexes has a complexity in $\mathcal{O}(n^4)$[5].

Let us assume that there are only ν variables that take can a duplicate value. We can chose B in order that the indexes of these variables are placed at the end of the list. Then all the symmetries in $\mathcal{G} \setminus \mathcal{G}^{[\beta_{n-\nu}]}$ can be broken with a linear number of constraints as in injective problems. Lexicographic constraints (4) are only required for symmetries that belong to $\mathcal{G}^{[\beta_{n-\nu}]}$. All the scopes of these constraints are included in a set of ν variables. Hence there are at most $\binom{\nu}{\mu}$ and the total number of constraints is in $\mathcal{O}(\binom{\nu}{\mu} + n)$. This prove the following:

Lemma 3. *Given a CSP where no more than μ simultaneous variables can be equal, and only a subset of ν variables can take duplicates values, all variable symmetries can be broken with a number of constraint which is FPT in ν and μ.*

equivalent to an inequality $x_{\beta_k} < x_{\sigma(\beta_k)}$ with $\sigma(\beta_k) \in \Delta^{[\beta_k]}$. Hence all variable symmetries can be broken with the following constraints:

$$\forall \beta_i \in B,\ \forall \gamma \in \Delta^{[\beta_i]}, \gamma \neq \beta_i,\ \text{we post}\ \ x_{\beta_i} < x_\gamma \tag{6}$$

The total number of inequalities is $\sum_{\beta_i \in B}(|\Delta^{[\beta_i]}| - 1)$ so it is in $\mathcal{O}(n\, log_2|\mathcal{G}|)$ or in $\mathcal{O}(n^2)$.

In [11], Puget has shown that equation (2) can be reduced to one inequality for each γ in $\Delta^{[\beta_i]}$. The principle is to associate each γ to the larger β_i (different from γ) such that $\gamma \in \Delta^{[\beta_i]}$.

Formally, let $\mathcal{R}_\gamma = \{\beta_i \in B \setminus \{\gamma\} \mid \gamma \in \Delta^{[\beta_i]}\}$. If $\mathcal{R}_\gamma \neq \emptyset$ let us define $r(\gamma) = \max(\mathcal{R}_\gamma)$ otherwise $r(\gamma) = \gamma$.

One can prove[6] that equation (2) is equivalent to the following linear number of constraints:

$$\forall \gamma \in 1..n \text{ such that } r(\gamma) \neq \gamma,\ \text{we post}\ \ x_{r(\gamma)} < x_\gamma \tag{7}$$

The case of heterogeneous domains

In Section 4, we provide a theoretical bound of number of constraints to be posted. As shown in example 1, this bound does not take into account initial domains of variables, when they are not all equal.

Example 1. Let $x_{i^1_\sigma}$ and $x_{\sigma(i^1_\sigma)}$ be two variables involved in a symmetry. If $D(x_{i^1_\sigma}) \cap D(x_{\sigma(i^1_\sigma)}) = \emptyset$, the constraint 5b is useless and can be discarded (not posted) as well as all constraints containing $x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)}$ in their left part.

The conjunction of the `Gcc` constraint and the initial domains of variables may also forbid combinations of equalities between pairs of variables:

Example 2. Let $D(x_1) \cap D(x_2) = \{v\} = D(x_3) \cap D(x_4)$ and $ub(v) = 3$. The `Gcc` forbids to have simultaneously $x_1 = x_2$ and $x_3 = x_4$. Then, all constraints 5c, within $x_1 = x_2 \wedge x_3 = x_4$ in their left part may be discarded.

More generally, during the generation of symmetry breaking constraints of the constraint network $N =< X, D, C >$, we propose the following technique: Before posting the constraint $x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)} \wedge \ldots \wedge x_{i^{\rho-1}_\sigma} = x_{\sigma(i^{\rho-1}_\sigma)} \rightarrow x_{i^\rho_\sigma} \leq x_{\sigma(i^\rho_\sigma)}$ we propose to solve the subproblem $N =< X, D, C' >$ with $C' = \{x_{i^1_\sigma} = x_{\sigma(i^1_\sigma)}, \ldots, x_{i^{\rho-1}_\sigma} = x_{\sigma(i^{\rho-1}_\sigma)}\} \cup \{Gcc\}$, the sub-problem of N restricted to both the `Gcc` constraint and the equality constraints of the left part of this constraint. If N' is not soluble, the constraint is discarded.

Lemma 4. *The satisfiability problem of the class of constraint networks involving only a* `Gcc` *and some equality constraints is polynomial.*

The idea of the proof relies on a slight modification of the `Gcc` encoding as flow problem. For each pair of variable (x_i, x_j) involved in an equality constraint,

[6] by transitivity, $x_{\beta_i} < x_\gamma$ is derived from $x_{\beta_i = r(\ldots r(r(\gamma)))} < \ldots < x_{r(r(\gamma))} < x_{r(\gamma)} < x_\gamma$

we replace them by a merged node $x_{i,j}$ The flow between the source s and this new node variable $x_{i,j}$ is exactly 2 to enforce both x_i and x_j to be assigned. The flow between $x_{i,j}$ and any value v in $D(x_i) \cap D(x_j)$ is either 0 or 2 to count 2 uses of v when x_{ij} (in fact both x_i and x_j) is assigned to v.

In addition of theoretical results in terms parametrized complexity, this technique provides a practical way to restrict the number of constraints to be posted for breaking symmetries in almost injective problems.

5 Conclusion

We have introduced a characterization of "almost injective" problems which is based on the number μ of variables that can be equal simultaneously. We showed that variable symmetry 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.

In case of heterogeneous domains, we presented a polynomial method (based on `Gcc`) for eliminating unnecessary constraints.

References

1. Benoist, T., Gaudin, E., Rottembourg, B.: Constraint programming contribution to benders decomposition: A case study. In: Proc. CP'02. pp. 603–617 (2002)
2. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C.G., Walsh, T.: The parameterized complexity of global constraints. In: Proc. AAAI. pp. 235–240 (2008)
3. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Filtering algorithms for the nvalue constraint. In: CPAIOR. pp. 79–93 (2005)
4. Bockmayr, A., Pisaruk, N., Aggoun, A.: Network flow problems in constraint programming. In: Proc. CP'01. pp. 196–210 (2001)
5. Butler, G.: Fundamental Algorithms for Permutation Groups, LNCS, vol. 559 (1991)
6. Crawford, J., Ginsberg, M., Luks, E., Roy, A.: Symmetry-breaking predicates for search problems. In: Proc. KR'96. pp. 148–159 (1996)
7. Downey, R.G., Fellows, M.R., Stege, U.: Parameterized complexity: A framework for systematically confronting computational intractability. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science. vol. 49, pp. 49–99 (1997)
8. Gent, I.P., Harvey, W., Kelsey, T.: Groups and constraints: Symmetry breaking during search. In: Proc. CP'02. LNCS, vol. 2470, pp. 415–430 (2002)
9. Gottlob, G., Szeider, S.: Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. Comput. J. 51(3), 303–325 (2008)
10. McKay, B.D.: *nauty user's guide*. Tech. rep., Australian National University, `http://cs.anu.edu.au/~bdm/nauty/` (2009)
11. Puget, J.F.: Breaking symmetries in all different problems. In: Proc. IJCAI'05. pp. 272–277 (2005)
12. Régin, J.C.: Generalized arc consistency for global cardinality constraint. Proc. AAAI'96 pp. 209–215 (1996)
13. Seress, Á.: Permutation group algorithms. Cambridge tracts in mathematics, Cambridge University Press (2003)