Symmetry Breaking in Subgraph Isomorphism

Stéphane Zampelli[(1)], Yves Deville[(1)], Mohamed Réda Saïdi[(2)], Belaïd Benhamou[(2)]

[(1)] Université catholique de Louvain,
Department of Computing Science and Engineering,
2, Place Sainte-Barbe 1348 Louvain-la-Neuve (Belgium)
[(2)]Centre de Mathématique et d'Informatique
39, rue Joliot Curie - 13453 Marseille cedex 13, France

Abstract

The present work studies symmetry breaking for the subgraph isomorphism problem. This NP-Complete problem decides if a pattern graph is isomorphic to a subgraph of a target graph. The first part of the paper shows how to detect and break all variable and value global symmetries. The second part studies local symmetries, and shows that subgraphs of the initial instance allow to efficiently compute local variable and value symmetries. Experiments show that global symmetries are an efficient technique for subgraph isomorphism, and that limited local symmetries may be useful for difficult instances.

1 Introduction

A symmetry in a Constraint Satisfaction Problem (CSP) is a bijective function that preserves CSP structure and solutions. Symmetries are important because they induce symmetric subtrees in the search tree. If the instance has no solution, failure has to be proved for equivalent subtrees regarding symmetries. If the instance has solutions, many symmetric solutions will have to be enumerated in symmetric subtrees. The detection and breaking of symmetries can thus speed up the solving of a CSP. Symmetries arise naturally in graphs as automorphisms. However, although many graph problems have been tackled [Beldiceanu *et al.*, 2005] [Cambazard and Bourreau, 2004] [Sellman, 2003] and a computation domain for graphs has been defined [Dooms *et al.*, 2005], and despite the fact that symmetries and graphs are related, little has been done to investigate the use of symmetry breaking for graph problems in constraint programming.

This work aims at applying and extending symmetry techniques for subgraph isomorphism. We show how to detect and handle global variable and value symmetries as well as local symmetries.

2 Background and Definitions

Basic definitions for subgraph isomorphism and symmetries are introduced.

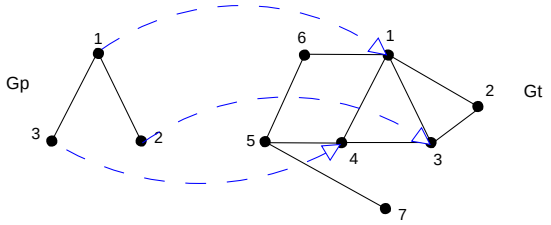

Figure 1: Example solution for an isomorphism problem instance.

A *graph* $G = (N, E)$ consists of a *node set* N and an *edge set* $E \subseteq N \times N$, where an edge (u, v) is a pair of nodes. The nodes u and v are the endpoints of the edge (u, v). We consider directed and undirected graphs. A *subgraph* of a graph $G = (N, E)$ is a graph $S = (N', E')$ where N' is a subset of N and E' is a subset of E such that for all $(u, v) \in E'$, $u, v \in N'$.

A *subgraph isomorphism problem* between a pattern graph $G_p = (N_p, E_p)$ and a target graph $G_t = (N_t, E_t)$ consists in deciding whether G_p is isomorphic to some subgraph of G_t. More precisely, one should find an injective function $f : N_p \rightarrow N_t$ such that $\forall (u, v) \in N_p \times N_p, (u, v) \in E_p \Rightarrow (f(u), f(v)) \in E_t$. This NP-Hard problem is also called subgraph monomorphism problem or subgraph matching in the literature. The function f is called a *subgraph matching function.*

The CSP model of subgraph isomorphism should represent a total function $f : N_p \rightarrow N_t$. This total function can be modeled with $X = x_1, \ldots, x_n$ with x_i a FD variable corresponding to the i^{th} node of G_p and $D(x_i) = N_t$. The injective condition is modeled with the global constraint $\texttt{alldiff}(x_1, \ldots, x_n)$. The isomorphism condition is translated into a set of constraints $MC_l(x_i, x_j) \equiv (x_i, x_j) \in E_t$ for all $(i, j) \in E_p$. This set of constraints can be turned into a global constraint $\texttt{MC}(x_1, \ldots, x_n) \equiv \bigwedge_{(i,j) \in E_p} MC_l(x_i, x_j)$. Implementation, comparison with dedicated algorithms, and extension to subgraph isomorphism and to graph and function computation domains can be found in [Zampelli *et al.*, 2005; Deville *et al.*, 2005].

A CSP instance is a triple $< X, D, C >$ where X is the set of variables, D is the universal domain spec-

ifying the possible values for those variables, and C is the set of constraints. In the sequel, $n = |N_p|$, $d = |D|$, and $D(x_i)$ is the domain of x_i. A symmetry over a CSP instance P is a bijection σ mapping solutions to solutions, and hence non solutions to non solutions [Puget, 2005b]. Since a symmetry is a bijection where domain and target sets are the same, a symmetry is a permutation. A *variable symmetry* is a bijective function $\sigma : X \rightarrow X$ permuting a (non) solution $s = ((x_1, d_1), \ldots, (x_n, d_n))$ to a (non) solution $\sigma s = ((\sigma(x_1), d_1), \ldots, (\sigma(x_n), d_n))$. A *value symmetry* is a bijective function $\sigma : D \rightarrow D$ permuting a (non) solution $s = ((x_1, d_1), \ldots, (x_n, d_n))$ to a (non) solution $\sigma s = ((x_1, \sigma(d_1)), \ldots, (x_n, \sigma(d_n))$. A *value and variable symmetry* is a bijective function $\sigma : X \times D \rightarrow X \times D$ permuting a (non) solution $s = ((x_1, d_1), \ldots, (x_n, d_n))$ to a (non) solution $\sigma s = (\sigma(x_1, d_1), \ldots, \sigma(x_n, d_n))$. A *global symmetry* of a CSP is a symmetry holding on the initial problem. A *local symmetry* of a CSP P is a symmetry holding only in a sub-problem $P^{'}$ of P. The conditions of the symmetry are the constraints necessary to generate $P^{'}$ from P [Gent *et al.*, 2005] [Benhamou, 1994]. A *group* is a finite or infinite set of elements together with a binary operation (called the group operation) that satisfies the four fundamental properties of closure, associativity, the identity property, and the inverse property. An *automorphism of a graph* is a graph isomorphism with itself. The set of automorphisms $Aut(G)$ defines a finite group of permutations.

3 Variable Symmetries

In this section, we show that the set of global variable symmetries of a subgraph isomorphism CSP is the set of automorphisms of the pattern graph. Moreover, we show how existing techniques can be used to break all global variable symmetries.

3.1 Detection

This subsection shows that, in subgraph isomorphism, global variable symmetries are the automorphisms of the pattern graph and do not depend on the target graph. It has been shown that the set of variable symmetries of the CSP is the automorphism group of a *symbolic graph* [Puget, 2005b]. The pattern G_p is transformed into a symbolic graph $S(G_p)$ where $Aut(S(G_p))$ is the set of variable symmetries of the CSP.

A CSP P modeling a subgraph isomorphism instance (G_p, G_t) can be transformed into the following symbolic graph $S(P)$:

1. Each variable x_i is a distinct node labelled i.
2. If there exists a constraint $MC(x_i, x_j)$, then there exists an arc between i and j in the symbolic graph.
3. The constraint alldiff is transformed into a node typed with label 'a'; an arc (a, x_i) is added to the symbolic graph for each x_i.

Figure 2 shows a pattern transformed into its symbolic graph. If we do not consider the extra node and

Figure 2: Example of symbolic graph for a square pattern.

arcs introduced by the alldiff constraint, then the symbolic graph $S(P)$ and G_p are isomorphic by construction. Given the labelling of nodes representing constraints, an automorphism in $S(P)$ maps the alldiff node to itself and the nodes corresponding to the variables to another node corresponding to the variables. Each automorphism in $Aut(G_p)$ will thus be a restriction of an automorphism in $Aut(S(P))$, and an element in $Aut(S(P))$ will be an extension of an element in $Aut(G_p)$. Hence the two following theorems.

Let (G_p, G_t) be a subgraph isomorphism instance, P its associated CSP. We have :

- $\forall\ \sigma \in Aut(G_p)\ \exists\ \sigma^{'} \in Aut(S(P)) :\ \forall\ n \in N_p :\ \sigma(n) = \sigma^{'}(n)$
- $\forall\ \sigma^{'} \in Aut(S(P))\ \exists\ \sigma \in Aut(G_p) :\ \forall\ n \in N_p :\ \sigma(n) = \sigma^{'}(n)$

Let (G_p, G_t) be a subgraph isomorphism instance, P its associated CSP. The set of variable symmetries of P is the set of bijective functions $Aut(S(P))$ restricted to N_p, which is equal to $Aut(G_p)$.

The above theorem states that only $Aut(G_p)$ has to be computed in order to get all variable symmetries.

3.2 Breaking

Two existing techniques are relevant to our particular problem. The first technique is an approximation and consists in breaking only the generators of the symmetry group [Crawford *et al.*, 1996]. Those generators are obtained by an automorphism detection software such as NAUTY[McKay, 1981] . For each generator σ, an ordering constraint $s \leq \sigma s$ is posted.

The second technique breaks all variable symmetries of an injective problem by using a Schreier-Sims algorithm, provided that the generators of the variable symmetry group are known [Puget, 2005a]. Puget showed that the number of constraints to be posted is linear with the number of variables. The Schreier-Sims algorithm computes a base and a strong generating set of a permutation group. Let G be the group, S_g the symmetry group of g elements containing G, and t the number of generators, then its complexity is in $O(g^2 log^3|G| + t.g.log|G|)$.

4 Value Symmetries

In this section we show how all global value symmetries can be detected and how existing techniques can be extended to break them.

4.1 Detection

In subgraph isomorphism, global value symmetries are automorphisms of the target graph and do not depend on the pattern graph.

Let (G_p, G_t) be a subgraph isomorphism instance and P be its associated CSP. Then each $\sigma \in Aut(G_t)$ is a value symmetry of P.

Proof Suppose that f is a subgraph isomorphism between G_p and G_t, and $f(i) = v_i$ for $i \in N_p$. Consider the subgraph $G = (N, E)$ of G_t, where $N = \{v_1, \ldots, v_n\}$ and $E = \{(i,j) \in E_t \mid (f^{-1}(i), f^{-1}(j)) \in E_p\}$. This means that there exists a isomorphic function f' matching G_p to σG. Hence $((x_1, \sigma(v_1)), \ldots, (x_n, \sigma(v_n)))$ is a solution. ■

4.2 Breaking

Breaking global value symmetries can be performed by using the GE-Tree technique [Ronay-Dougal *et al.*, 2004]. The idea is to modify the distribution by avoiding symmetrical value assignments. Suppose a state S is reached, where $x_1, \ldots, x_k$ are assigned to $v_1, \ldots, v_k$ respectively, and $x_{k+1}, \ldots, x_n$ are not assigned yet. The variable x_{k+1} should not be assigned to two symmetrical values, since two symmetric subtrees would be searched. For each value $v_i \in D(x_{k+1})$ that is symmetric to a value $v_j \in D(x_{k+1})$, only one state S_1 should be generated with the new constraint $x_{k+1} = v_i$.

A convenient way to compute those symmetrical values uses the Schreier-Sims algorithm. Algorithm Schreier-Sims outputs the sets $U_i = \{k \mid \exists\, \sigma \in Aut(G_t) : \sigma(i) = k \wedge \sigma(j) = j\ \forall\, j < i\}$. A set U_i gives the images of i by the automorphisms of G mapping $0, \ldots, i-1$ to themselves. If values are assigned in an increasing order, assigning symmetrical values can be avoided by using those sets U_i. Using symmetry breaking constraints together with GE-Tree is complete and correct as shown in [Puget, 2005a].

5 Local Symmetries

Global symmetries may hide symmetries arising during search. During search, variables are assigned and new variable symmetries arise. As values are removed from domains, new value symmetries are created and can be exploited. In this section, we focus on detecting those symmetries for the subgraph isomorphism problem.

Local symmetries for subgraph isomorphism can be found through local graphs of the initial problem. During the search, subgraphs of the pattern and target graph define variable and value local symmetries. We first show how to define those subgraphs, and then we explain local variable symmetry detection and local value symmetry detection.

5.1 Partial dynamic graphs

We first introduce partial dynamic graphs. Those graphs are associated to a state in the search and correspond to the unsolved part of the problem. This can be viewed as a new local problem to the current state.

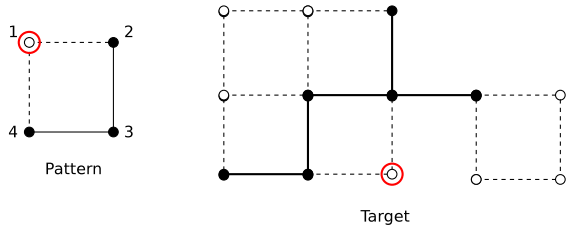

Figure 3: Example of local subgraphs.

Let S be a state in the search.
The **partial dynamic pattern graph** $G_p^- = (N_p^-, E_p^-)$ induced by S is a subgraph of G_p such that :

- $N_p^- = \{i \in N_p \mid \exists\, j : (i,j) \in E_p \wedge \exists\, a \in D(x_i) \wedge \exists\, b \in D(x_j) \wedge (a,b) \notin E_t\}$
- $E_p^- = \{(i,j) \in E_p \mid i \in N_p^- \wedge j \in N_p^-\}$

The **partial dynamic target graph** $G_t^- = (N_t^-, E_t^-)$ is a subgraph of G_t such that :

- $N_t^- = \cup_{i \in N_p^-} D(x_i)$
- $E_t^- = \{(a,b) \in E_t \mid a \in N_t^- \wedge b \in N_t^-\}$

Those partial dynamic graphs define the local CSP corresponding to the local state.

Figure 3 shows an example where circled nodes are assigned to each other. In the pattern graph, plain nodes and edges represent G_p^-. Regarding morphism constraints, dashed edges are entailed MC_l constraints and plain edges are non entailed MC_l constraints. In the target graph, plain nodes and edges represent G_t^- assuming a forward checking propagation for the MC_l constraints.

One general way to compute local symmetries is to use the microstructure of the CSP [Cohen *et al.*, 2006]. The set of nodes of the microstructure graph is the product set of the variables and the domain. In our particular problem of subgraph isomorphism, the variables are the nodes of G_p^- and the domain is the set of nodes of G_t^-. Hence the size of the microstructure is $|G_p^- \times G_t^-|$ and can be very large. But in subgraph isomorphism, local symmetries can be computed directly in the graphs G_p^- and G_t^-, without using the microstructure.

5.2 Local variable symmetries

Local variable symmetries must map variables having the same domain. This fact follows directly from the definition of a variable symmetry. This problem was not present for global variable symmetries as the initial domains are N_t. The set of automorphisms of the partial dynamic pattern graph has to be redefined.

Given a partial dynamic pattern graph G_p^-, $Aut'(G_p^-)$ is the set of automorphisms mapping a node i to a node j if and only if $D(x_i) = D(x_j)$. The following theorem states that local variable symmetries can be obtained by computing $Aut'(G_p^-)$.

Let (G_p, G_t) be a subgraph isomorphism instance, L be a state in the search space, G_p^- the partial dynamic

pattern graph associated with L, and P' be the CSP associated with L.Then each $\sigma \in Aut'(G_p^-)$ is a variable symmetry of $P^{'}$.

Proof Let $\sigma \in Aut'(G_p^-)$. Consider the symbolic graph $Aut(S(P'))$ of P'. Recall that the alldiff constraint has no influence on $Aut(S(P'))$. All automorphisms β of $Aut(S(P'))$ are not variable symmetry of P' since domains of variables may be different in the local subproblem P'. Since $\sigma \in Aut(S(P'))$ and is restricted to map only variables with the same domains, σ is a variable symmetry of P'. ■

Computing $Aut'(G_p^-)$ can be done as usual by using automorphism detection software. The initial partition is refined into ordered sets containing variables having the same domain.

Breaking

Local variable symmetries can be broken by using the same technique for global variable symmetries (Section 3.2). This ensures that all detected local variable symmetries are broken. However, adding breaking constraint of the form $x_i < x_j$ modify the local symbolic graph $S(P)$. This may introduce or remove new local variable symmetries. The detection presented in the previous section is however valid. Indeed, the additional constraints $x_i < x_j$ ensure that $D(x_i) \neq D(x_j)$. Any automorphism between x_i and x_j is excluded from $Aut'(G_p^-)$.

5.3 Local value symmetries

The following theorem states that value symmetries of the local CSP P' can be obtained by computing $Aut(G_t^-)$ and that these symmetries can be exploited without losing or adding solutions to the initial problem.

Let (G_p, G_t) be a subgraph isomorphism instance, P' be the local CSP associated with a state during the search. Then each $\sigma \in Aut(G_t^-)$ is a value symmetry of $P^{'}$. **Proof** This follows directly from Theorem 4.1 and the fact that (G_p^-, G_t^-) is a subgraph isomorphism instance. ■

The dynamic target graph G can be computed dynamically. In [Deville *et al.*, 2005], we showed how subgraph isomorphism can be modeled and implemented in CP(Graph), an extension of CP with graph domain variables [Dooms *et al.*, 2005]. The domain of a graph variable is modeled by a lower bound and an upper bound graph, and represents all the graphs between the lower and upper bound. In this setting, a graph domain variable T represents the matched target subgraph. The initial lower bound of T is the empty graph, and the initial upper bound if G_t. When a solution is found, T is instantiated to the matched subgraph of G_t. Hence, during the search, the dynamic target graph G_t^- will be the upper bound of variable T and can be obtained in $O(1)$.

Speeding up detection

Computing directly $Aut(G_t^-)$ is correct but this computation can be fasten. Actually, all value symmetries are not possible in a local instance (G_p^-, G_t^-). Only nodes that are all present in at least one domain can be mapped to each other in a value symmetry of P'. The search tree of the automorphism algorithm can be pruned when such nodes are mapped together.

Breaking

In this subsection, we show how to modify the GE-Tree method to handle local value symmetries. Before distribution, the following actions are triggered :

1. Compute the partial dynamic target graph G_t^-.
2. The NAUTY and Schreier-Sims algorithms are called to produce the new $U_i^{'}$ sets.
3. Given a state S, a new variable and value selection can be used such that local value symmetries are broken :
   (a) a new state S_1 with a constraint $x_k = v_k$
   (b) a new state S_2 with constraints : $x_k \neq v_k$ and $x_k \neq v_j \ \forall \ j \in U_{k-1} \cup U_{k-1}^{'}$.

The only difference with the original GE-Tree method is the addition of the U'_{k-1} during the creation of the second branch corresponding to the state S_2.

An issue is how to handle the global and local structures U. In the Gecode system (http://www.gecode.org), in which the actual implementation is made, the states are copied and trailing is not needed. Thus the global structure U must not be updated because of backtracking. A single global copy is kept during the whole search process. In a state S where local values symmetries are discovered, structure U is copied into a new structure $U^{''}$ and merged with $U^{'}$. This structure $U^{''}$ shall be used for all states $S^{'}$ having S in its predecessors.

6 Experimental results

The objectives in this section are to assess performances of global symmetries, and performance of local symmetries against global symmetries. For local symmetries, we study the overhead of computing local symmetry information and their ability to solve more difficult instances. Moreover, we would like to know whether local symmetries can be applied on the whole search space.

The CSP model for subgraph isomorphism has been implemented in Gecode, using CP(Graph) and CP(Map) [Dooms *et al.*, 2005] [Deville *et al.*, 2005] . The CP(Graph) framework provides graph domain variables and CP(Map) provides function domain variables. All the software is implemented in C++. The standard implementation of NAUTY [McKay, 1981] algorithm is used. We also implemented Schreier-Sims algorithm. The computation of the constraints for breaking injective problems is implemented, and GE-Tree method is also incorporated. All local symmetry techniques presented are also implemented.

Instances - The data graphs used to generate instances are from the GraphBase database containing different topologies and has been used in [Larrosa and Valiente, 2002]. Experiments are performed on the first

50 undirected graphs from GraphBase. The undirected set was selected because it holds potentially more symmetries than the directed graphs. This undirected set contains graphs ranging from 10 nodes to 138 nodes. All those graphs are tested for isomorphism with one another. Only subgraph isomorphism instances with a pattern graph smaller than the target graph are kept. There are 1225 instances.

Setup - All runs were performed on a dual Intel(R) Xeon(TM) CPU 2.66GHz with 2 Go of RAM. In our tests, we look for all solutions. This ensures that we measure the whole tree search reduction, and we avoid strong influence of the heuristic. As shown later in this section, the number of solved instances stabilizes for all instances after a couple of minutes. Hence a run time limit is set. A run is solved if it finishes in less than 5 minutes, unsolved otherwise. Detecting the local symmetries on the whole search space tends to be time-consuming. Hence local symmetry detection is seen as an extension of global symmetries. No detection is made when 3 variables are instantiated. Breaking is performed over the whole search space.

Automorphism detection time - A main concern is how much time it takes to compute the symmetries of the graphs. Regarding global symmetries, NAUTY processed each undirected graph in less than 0.02 second. All undirected graphs were processed by Schreier-Sims in less than one second, except two of them, with 4 seconds and 8 seconds. This shows a negligible time regarding symmetry detection on this set of instances.

Models - Depending on the symmetry breaking techniques, various models are selected for these experiments :

- vflib : state of the art dedicated C++ algorithm [Cordella *et al.*, 2001]
- light : simple CP model
  - Forward checking constraints
  - No redundant constraint
- heavy : advanced CP model
  - Arc consistency
  - Redundant constraint [Larrosa and Valiente, 2002]
- global var : heavy + global variable symmetry
- global value : heavy + global value symmetry
- global varvalue : heavy + global variable and value symmetry
- local var : heavy + local variable symmetry
- local value : heavy + local value symmetry
- glocal var : heavy + global *and* local variable symmetry
- glocal value : heavy + global *and* local value symmetry

Vflib and the light model are considered as basic models since they perform only forward checking. We call easy instances those instances that are quickly solved by vflib and the light model. Those instances do not require any arc consistent or redundant constraint.

Detailed results - We study first experimental results for global symmetries. Figure 4 shows the number of solved instances against time. This Figure justifies the choice of a time limit of 5 minutes, as most of the solved instances are solved during the first 100 seconds. Hence only the percentage of solved instances is relevant. Figure 5 shows the detailed results. The total time is the time to solve all instances, the mean time is the mean time over all solved instances, the common mean time(memory) is the mean time(memory) over instances solved by vflib. Global symmetries clearly outperforms light, heavy and vflib and improve time on easy instances and all instances. Thanks to global variable and value symmetries, 18% more instances are solved compared to vflib and all instances are solved much more efficiently.

We now study the experimental results for local symmetries. Figure 6 shows the detailed results. The common time is still reduced, but local symmetries achieve the same performance as the heavy model without any symmetry technique, with the exception of local and global variable symmetries. Those results for local symmetries are due to the time needed to compute local symmetries. Actually, some easy instances are not solved with local symmetries.

In order to assess efficiency of local symmetries for difficult problems, we performed the following experiment. The light model is ran for 30 seconds, and if the instance is not solved, local symmetry models are used for 270 seconds. This coroutining setup ensures that easy instances are solved. Results for this new setup are shown in Figure 7. We compare the results of local symmetries against global symmetries. Local symmetries slightly outperform global symmetries. Inside local symmetry models, the models combining global symmetries outperform pure local symmetry models. This is mainly because some instances contains a lot of symmetries that disappear during search. Local symmetry has a high cost but reduces time for difficult instances. Not surprisingly, local symmetries performance are poor on easy instances, but outperform global symmetry on difficult instances.

To the best of our knowledge, subgraph isomorphism with symmetry breaking achieves the best percentage of solved instances over the GraphBase benchmark proposed by [Larrosa and Valiente, 2002].

7 Conclusion

In subgraph isomorphism, both global variable and value symmetries can be computed on the initial instance. Indeed, this computation can be made directly on the pattern graph and the target graph. Moreover, all variable and value symmetries can be broken by computing a base and a strong generating set of the permutation groups thanks to the Schreier-Sims algorithm. Local variable and value symmetries can be found in a similar way. A

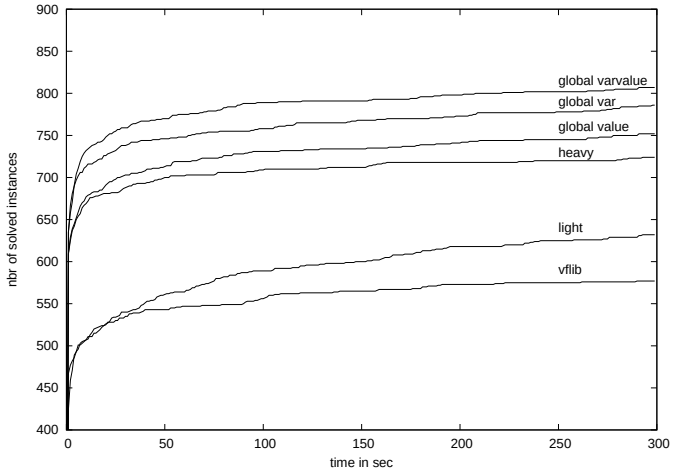

Figure 4: Results for global symmetries

	solved	total time	mean time	mean mem	c. mean time	c. mean mem.
vflib	47.1%	3329 min.	9.35 sec.	115 kb	9.35 sec.	92 kb
light	51.4%	3162 min.	17.91 sec.	2028 kb	14.89 sec.	1391 kb
heavy	58.94%	2584 min.	6.57 sec.	7892 kb	5.81 sec.	3103 kb
global var	64%	2318 min.	8.72 sec.	7744 kb	2.75 sec.	2933 kb
global value	61.2%	2479 min.	8.45 sec.	7820 kb	3.06 sec.	3014 kb
global varvalue	65.7%	2197 min.	7.35 sec.	7545 kb	2.50 sec.	2983 kb

Figure 5: Detailed results for global symmetries.

	solved	total time	mean time	mean mem	c. mean time	c. mean mem.
heavy	58.94%	2584 min.	6.57 sec.	7892 kb	5.81 sec.	3103 kb
glocal var	60.82%	2473 min.	5.93 sec.	19201 kb	4.40 sec.	3182 kb
local var	58.45%	2615 min.	5.88 sec.	18666 kb	3.30 sec.	3096 kb
glocal value	58.78%	2601 min.	6.37 sec.	17839 kb	3.88 sec.	3684 kb
local value	57.14%	2682 min.	4.96 sec.	7812 kb	3.29 sec.	3243 kb

Figure 6: Detailed results for local symmetries over all instances.

	solved	total time	mean time	mean mem
global var	64,73%	2203 min.	4.27 sec.	11371 kb
glocal var	65,96%	2150 min.	3.26 sec.	11503 kb
local var	63,10%	2300 min.	3.15 sec.	4105 kb
global value	63,92%	2256 min.	2.94 sec.	11475 kb
glocal value	64,49%	2234 min.	3.78 sec.	28610 kb
local value	63,18%	2301 min.	3.77 sec.	4330 kb

Figure 7: Detailed results for local symmetries with coroutining.

suitable definition of the local pattern and target graphs makes the computation of local symmetries as direct as for global symmetries.

Experimental results suggest that breaking all variable and value symmetries is an efficient way to solve difficult instances. Global symmetries together with arc consistency and redundant constraints were able to solve 65% of the instances, which makes constraint programming the most efficient technique for this data set. Local symmetries achieve also good results on difficult instances. However, computing local symmetries may not be the good tradeoff between search and symmetries, especially for easy instances. Computing local symmetries during the whole search is inefficient.

Interesting directions include experiments on faster but weaker detection methods. One could search for and break generators, as the Schreier-Sims tends to be time consuming. Other weaker forms of detection could also be used. Finally, experiments should be conducted on real-world class of graphs such as scale-free networks.

Acknowledgments

The authors want to thank the anonymous reviewers for the helpful comments. This research is supported by the Walloon Region, project Transmaze (WIST516207) and by the Interuniversity Attraction Poles Programme (Belgian State, Belgian Science Policy).

References

[Beldiceanu *et al.*, 2005] Nicolas Beldiceanu, Pierre Flener, and Xavier Lorca. The tree constraint. In Roman Bartak, editor, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Second International Conference, CPAIOR 2005, Prague, Czech Republic, Proceedings*, volume 3524 of *Lecture Notes in Computer Science*, pages 64–78. Springer, June 2005.

[Benhamou, 1994] Belaid Benhamou. Study of symmetry in constraint satisfaction problems. In Alan Borning, editor, *Second International Workshop on Principles and Practice of Constraint Programming, PPCP'94, Proceedings*, volume 874 of *Lecture Notes in Computer Science*, pages 246–254, Orcas Island, Seattle, USA, may 1994. Springer.

[Cambazard and Bourreau, 2004] Hadrien Cambazard and Eric Bourreau. Conception d'une contrainte globale de chemin. In *10e Journées nationales sur la résolution pratique de problèmes NP-complets (JNPC'04)*, pages 107–121, Angers, France, June 2004.

[Cohen *et al.*, 2006] David Cohen, Peter Jeavons, Christopher Jefferson, Karen E. Petrie, and Barbara M. Smith. Symmetry definitions for constraint satisfaction problems. *Constraints*, 11(2-3):115–137, 2006.

[Cordella *et al.*, 2001] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 149–159. Cuen, 2001.

[Crawford *et al.*, 1996] James Crawford, Matthew L. Ginsberg, Eugene Luck, and Amitabha Roy. Symmetry-breaking predicates for search problems. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *KR'96: Principles of Knowledge Representation and Reasoning*, pages 148–159. Morgan Kaufmann, San Francisco, California, November 1996.

[Deville *et al.*, 2005] Yves Deville, Grégoire Dooms, Stéphane Zampelli, and Pierre Dupont. Cp(graph+map) for approximate graph matching. In Francisco Azevedo, Carmen Gervet, and Enrico Pontelli, editors, *1st International Workshop on Constraint Programming Beyond Finite Integer Domains (in conjunction with the Eleventh International Conference on Principles and Practice of Constraint Programming), CP2005, Sitges, Spain*, pages 33–47, October 2005.

[Dooms *et al.*, 2005] Grégoire Dooms, Yves Deville, and Pierre Dupont. Cp(graph): Introducing a graph computation domain in constraint programming. In van Beek [2005], pages 211–215.

[Gent *et al.*, 2005] Ian .P. Gent, Tom Kelsey, Steve A. Linton, Iain McDonald, Ian Miguel, and Barbara M. Smith. Conditional symmetry breaking. In van Beek [2005], pages 256–270.

[Larrosa and Valiente, 2002] Javier Larrosa and Gabriel Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical. Structures in Comp. Sci.*, 12(4):403–422, 2002.

[McKay, 1981] B. D. McKay. Pratical graph isomorphism. *Congressum Numerantium*, 30:35–87, 1981.

[Puget, 2005a] Jean-Francois Puget. Breaking symmetries in all different problems. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI*, pages 272–277. Professional Book Center, 2005.

[Puget, 2005b] Jean-François Puget. Automatic detection of variable and value symmetries. In van Beek [2005], pages 477–489.

[Ronay-Dougal *et al.*, 2004] C.M. Ronay-Dougal, I.P. Gent, T. Kelsey, and S. Linton. Tractable symmetry breaking in using restricted search trees. In Ramon Lópes de Mántaras and Lorenza Saitta, editors, *16th Europen Conference on Artificial Intelligence, Proceedings, Valencia, Spain*, volume 110, pages 211–215. IOS Press, 2004.

[Sellman, 2003] M. Sellman. Cost-based filtering for shorter path constraints. In Fransceca Rossi, editor, *Nineth International Conference on Principles and Pratice of Constraint Programming, CP 2003, Kinsale, Ireland, Proceedings*, volume 2833 of *Lecture Notes in Computer Science*, pages 694–708. Springer-Verlag, october 2003.

[van Beek, 2005] Peter van Beek, editor. *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, Lecture Notes in Computer Science, Barcelona, Spain, October 2005. Springer.

[Zampelli *et al.*, 2005] Stéphane Zampelli, Yves Deville, and Pierre Dupont. Approximate constrained subgraph matching. In van Beek [2005], pages 832–836.