Future and trends of Constraint Programming

Frédéric BENHAMOU
Narendra JUSSIEN
Barry O'SULLIVAN

December 1, 2006

Contents

Chapter 1

Symmetry Breaking in Subgraph Pattern Matching

1.1. 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 a lot of graph problems have been tackled [BEL 05] [CAM 04] [SEL 03] and a computation domain for graphs has been defined [DOO 05], 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 matching. We show how to detect and handle global variable and value symmetries as well as local value symmetries.

Related Works Handling symmetries to reduce search space has been a subject of research in constraint programming for many years. Crawford and al. [CRA 96] showed that computing the set of predicates breaking the symmetries of an instance

Chapter written by Zampelli STÉPHANE and Deville YVES and Dupont PIERRE.

is NP-hard in general. Different approaches exist for exploiting symmetries. Symmetries can be broken during search either by posting additional constraints (SBDS) [GEN 01b] or by pruning the tree below a state symmetrical to a previous one (SBDD) [GEN 03]. Symmetries can be broken by taking into account the symmetries into the heuristic [MES 01]. Symmetries can be broken by adding constraints to the initial problem at its root node [CRA 96] [GEN 01a]. Symmetries can also be broken by remodelling the problem [SMI 01].

Dynamic detection of value symmetries (also called local value symmetries or conditional value symmetries) and a general method for detecting them has been proposed in [BEN 94]. The general case for such a detection is difficult. However in not-equal binary CSPs, some value symmetries can be detected in linear time [BEN 04] and dominance detection for value symmetries can be performed in linear time [BEN 06].

Lately research efforts has been triggered towards defining, detecting and breaking symmetries. Cohen and al. [COH 06] defined two types of symmetries, solution symmetries and constraint symmetries and proved that the group of constraint symmetries is a subgroup of solution symmetries. Gent and al. [GEN 05b] rediscovered local symmetries defined in [BEN 94] and evaluated several techniques to break local symmetries. However the detection of local symmetries remains a research topic. Symmetries were also shown to produce stronger forms of consistency and more efficient mechanisms for establishing them [GEN 05a]. Finally, Puget [PUG 05b] showed how to detect symmetries automatically, and showed that all variable symmetries could be broken with a linear number of constraints for injective problems [PUG 05a].

Graph pattern matching is a central application in many fields [CON 04]. Many different types of algorithms have been proposed, ranging from general methods to specific algorithms for particular types of graphs. In constraint programming, several authors [LAR 02, RUD 98] have shown that subgraph matching can be formulated as a CSP problem, and argued that constraint programming could be a powerful tool to handle its combinatorial complexity. Within the CSP framework, a model for subgraph monomorphism has been proposed by Rudolf [RUD 98] and Valiente et al. [LAR 02]. Our modeling [ZAM 05] is based on these works. Sorlin and Solnon [SOR 04] proposed a filtering algorithm based on paths for graph isomorphism and part of our approach can be seen as a generalization of this filtering. The same authors recently proposed a new filtering algorithm for graph isomorphism based on iterative labelling of nodes using local neighborhood structure [SOR 06]. A declarative view of matching has also been proposed in [MAM 04]. In [ZAM 05], we showed that the CSP approach is competitive with dedicated algorithms over a graph database representing graphs with various topologies.

Objectives This work aims at developing symmetry breaking techniques for subgraph matching modelled as a CSP in order to increase the number of tractable instances of graph matching. Our first goal is to develop specific detection techniques

for the classical variable symmetries and value symmetries, and to break such symmetries when solving subgraph matching. Our second goal is to develop local symmetries detection and breaking techniques that can be easily handled for subgraph matching.

Results

– We show that all global variable symmetries can be detected by computing the set of automorphisms of the pattern graph, and how they can be broken.

– We show that all global value symmetries can be detected by computing the set of automorphisms of the target graph, and how they can be broken.

– Experimental results compare and analyze the enhancement achieved by global symmetries and show that symmetry breaking is an effective way to increase the number of tractable instances of the subgraph matching problem.

– We show that local value symmetries can be detected by computing the set of automorphisms on various subgraphs of the target graph. The GE-Tree method can be extended to handle these local symmetries.

Outline Sections 2 provides the necessary background in subgraph matching and in symmetry breaking. Section 3 describes a CSP approach for subgraph matching. Sections 3 and 4 present variable symmetries and value symmetries in subgraph matching. Section 5 describes experimental results for global symmetries. Local value symmetries are discussed in Section 6. Finally, Section 7 concludes this work.

1.2. Background and Definitions

Basic definitions for subgraph matching and symmetries are introduced.

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 monomorphism** (or subgraph matching) between G_p and G_t is a total injective function $f : N_p \rightarrow N_t$ respecting the monomorphism constraint : $(u, v) \in E_p \Rightarrow (f(u), f(v)) \in E_t$. Figure 1.1 shows an example of subgraph monomorphism.

The CSP model of subgraph matching 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 ith 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 monomorphism condition is translated into a set of constraints $MC_l(x_i, x_j) \equiv (x_i, x_j) \in E_t$

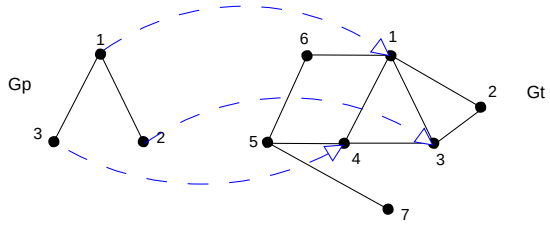

Figure 1.1. *Example solution for a monomorphism problem instance.*

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 [ZAM 05, DEV 05]. A CSP instance is a triple $< X, D, C >$ where X is the set of variables, D is the universal domain specifying 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 [PUG 05b]. 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 [GEN 05b] [BEN 94]. 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.

1.3. Variable Symmetries

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

1.3.1. *Detection*

This subsection shows that, in subgraph matching, 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* [PUG 05b]. 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.

DEFINITION.– *A CSP P modeling a subgraph monomorphism 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 1.2 shows a pattern transformed into its symbolic graph. If we do not consider the extra node and 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.

THEOREM.– *Suppose we have a subgraph monomorphism instance (G_p, G_t) and its associated CSP P. Then :*

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

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

THEOREM.– *Given a subgraph monomorphism instance (G_p, G_t) and its associated CSP P, 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.

1.3.2. *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

Figure 1.2. *Example of symbolic graph for a square pattern.*

group [CRA 96]. Those generators are obtained by using a tool such as NAUTY. 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 SchreierSims algorithm, provided that the generators of the variable symmetry group are known [PUG 05b]. Puget showed that the number of constraints to be posted is linear with the number of variables. The SchreierSims algorithm computes a base and a strong generating set of a permutation group in $O(g^2 log^3|G| + t.g.log|G|)$, where G is the group, t the number of generators and g the size of the group of all permutations containing G.

1.4. 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.

1.4.1. *Detection*

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

THEOREM.– *Given a subgraph monomorphism instance (G_p, G_t) and its associated CSP P, each $\sigma \in Aut(G_t)$ is a value symmetry of P.*
Proof Suppose that f is a subgraph monomorphism 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 monomorphic function $f^{'}$ matching G_p to σG. Hence $((x_1, \sigma(v_1)), \ldots, (x_n, \sigma(v_n)))$ is a solution. ■

1.4.2. *Breaking*

Breaking global value symmetries can be performed by using the GE-Tree technique [RON 04]. 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 SchreierSims algorithm. Algorithm SchreierSims 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 [PUG 05b].

1.5. Experimental results

This section presents experiments for global variable and value symmetries.

The CSP model for subgraph monomorphism has been implemented in Gecode (http://www.gecode.org), using CP(Graph) and CP(Map) [DOO 05] [DEV 05] . 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 [MCK 81] algorithm is used. We also implemented SchreierSims algorithm. The computation of the constraints for breaking injective problems is implemented, and GE-Tree method is also incorporated.

We have evaluated global variable symmetry detection and breaking, global value symmetry detection and breaking, and global variable and value symmetry breaking.

The data graphs used to generate instances are from the GraphBase database containing different topologies and has been used in [LAR 02]. There is a directed and an undirected set of graphs. Experiments are performed on undirected and directed graphs, because automorphism groups are expected to be larger in undirected graphs than in directed graphs. We took the first 30 directed graphs and the first 50 undirected graphs from GraphBase. The directed set contains graphs ranging from 10 nodes to 462 nodes. The undirected set contains graphs ranging from 10 nodes to 138 nodes. Using those graphs, there are 405 instances for directed graphs and 1225 instances for undirected graphs. All runs were performed on a dual Intel(R) Xeon(TM) CPU 2.66GHz.

A main concern is how much time it takes to compute the symmetries of the graphs. NAUTY processed each undirected graph in less than 0.02 second. For directed graphs, each graph is processed in less than 0.01 second except one of them which terminate in 0.8 second and 4 of them which did not terminate in five minutes. Note that we did not tune NAUTY. The SchreierSims algorithm computes the U structure for each directed graph in less than one second except for 3 of them which

Table 1.1. *Variable symmetries*

Undirected graphs				Directed graphs			
	solved	total time	mean time		solved	total time	mean time
CSP	58%	70 min.	5.95 sec.	CSP	67%	21 min.	4.31 sec.
Gen.	60,5%	172 min.	13.95 sec.	Gen.	74%	47 min.	8.87 sec.
FVS	61.8%	101 min.	8 sec.	FVS	74%	40 min.	7.64 sec.

Table 1.2. *Value Symmetries*

Undirected graphs				Directed graphs			
	solved	total time	mean time		solved	total time	mean time
CSP	53,7%	31 min.	20.1 sec.	CSP	67%	21 min.	4.31 sec.
GE-Tree	55,3%	6 min.	3.21 sec.	GE-Tree	68%	21 min.	4.39 sec.

terminate in 0.5 second, 1 of them in 1.5 seconds, and 1 of them in 3.1 seconds. All undirected graphs were processed by SchreierSims in less than one second, except two of them, with 4 seconds and 8 seconds.

In our tests, we look for all solutions. A run is solved if it finishes in less than 5 minutes, unsolved otherwise. We applied the basic CSP model, the model where constraints that break variable symmetries with generators (Gen.) are posted, and finally the full variable symmetry technique (FVS) that breaks all variable symmetries. Results are shown in Table 1.1. In those runs, the preprocessing time has not been considered. The total time column shows the total time needed for the solved instances. The mean time column shows the mean time for the solved instances.

Thanks to variable symmetry breaking constraints more instances are solved, for the directed graphs as well as for the undirected graphs. Moreover, the time for solved instances is increased because of the variable symmetry breaking constraints. Regarding the mean time, the full variable symmetry breaking constraint has a clear advantage.

Value symmetry breaking is evaluated on the set of directed graphs and undirected graphs. Table 1.2 shows that around one percent is gained. However the mean time for undirected graphs is decreased, even thought this is not the case for directed graphs. This may be due to the fact that there are less symmetries in directed graph than in undirected graphs. For variable and value symmetries, a total of 233 undirected random instances were treated. We evaluated variable and values symmetries separately and then together in Table 1.3. This table shows that, as expected, value symmetries and variable symmetries each increases the number of solved instances. Notice here that value symmetry breaking with GE-Tree leads to new solved instances and better performance, reducing mean time on solved instances. The full variable symmetry

Table 1.3. *Variable and value symmetries.*

Undirected graphs			
	solved	total time	mean time
CSP	53,7%	31 min.	20.1 sec.
GE-Tree	55,3%	6 min.	3.21 sec.
FVS	54,9 %	31 min.	19 sec.
GE-Tree and FVS	55,3 %	26 min.	8.68 sec.

technique makes new instances solved, but does not significantly reduce mean time on solved instances. Moreover, the combination of value symmetry breaking and variable symmetry breaking does not combine the power of the two techniques. In fact the GE-Tree upper bound of the number of the solved solutions is not increased by using full variable symmetry technique, and its mean time is even increased.

From these experiments, we conclude that global variable and value symmetry techniques give better performances and solve new instances. However they are not sufficient to solve a significant higher percentage of instances. The next section presents how to detect and handle local value symmetries.

1.6. Local Value Symmetries

In subgraph monomorphism, the relations between values are explicitly represented in the target graph. This allows the detection of local values symmetries. Consider Figure 1.3. Only global value symmetries of P are in $Aut(G_t)$. There exists at least two local value symmetric solutions : $\{(x_1, 1), (x_2, 2), (x_3, 3), (x_4, 4)\}$ and $\{(x_1, 2), (x_2, 1), (x_3, 4), (x_4, 3)\}$ although $Aut(G_t) = \emptyset$.

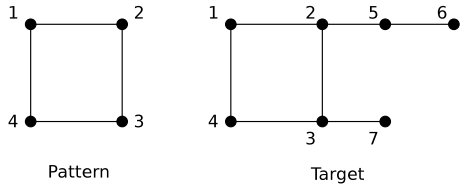

Figure 1.3. *Example of matching whithout value symmetries but with local value symmetries.*

Two techniques are presented in this section. The first one uses the target subgraph defined by the union of the current domains, called the dynamic target graph, and the second uses the graphs local to the current subproblem, called the partial dynamic graphs.

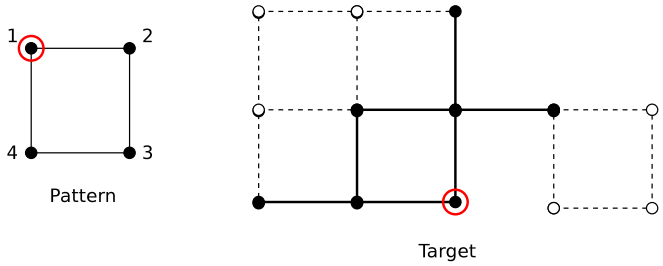

Figure 1.4. *Example of dynamic target subgraph.*

1.6.1. *Dynamic target graph*

This first technique to detect local value symmetries considers the subgraph of the union of the current variable domains.

1.6.1.1. *Detection*

During the search, the target graph looses a node a whenever $a \notin \cup_{i \in N_p} D(x_i)$. This is interesting because the relation between the values are known dynamically.

The set of values $\cup_{i \in N_p} D(x_i)$ denotes the nodes of subgraph of G_t in which a solution is searched. For a given state S, such a subgraph can be computed efficiently. We first define this subgraph of G_t.

DEFINITION.– *Let S be a state in the search where $x_1, \ldots, x_k$ are assigned, and $x_{k+1}, \ldots, x_n$ are not assigned. The **dynamic target graph** $G_t^* = (N_t^*, E_t^*)$ is a subgraph of G_t such that :*

– $N_t^* = \cup_{i \in [1,\ldots,n]} D(x_i)$

– $E_t^* = \{(a, b) \in E_t \mid a \in N_t^* \wedge b \in N_t^*\}$

Figure 1.4 shows an example of a dynamic target graph. In this figure, the circled nodes are mapped in the current assignement. The blank nodes are the nodes excluded from the union of the current domains, and the black nodes are the nodes included in this union. The plain edges are the selected edges for the dynamic target subgraph. The following theorem shows that each automorphism of G_t^* is a local value symmetry for the state S.

THEOREM.– *Suppose we have a subgraph monomorphism instance (G_p, G_t), its associated CSP P, and a state S in the search, each $\sigma \in Aut(G_t^*)$ is a local value symmetry of P. Moreover, the conditions of σ are $x_1 = v_1, \ldots, x_k = v_k$.*

Proof Suppose $Sol = (v_1, \ldots, v_k)$ is a partial solution. Consider the dynamic target

subgraph G_t^*. The state S can be considered as a new CSP $P^{'}$ of an instance (G_p, G_t^*) with additional constraints $x_1 = v_1, \ldots, x_k = v_k$. By Theorem from Section 1.4.1, the thesis follows. ■

The size of G_t^* is an important issue, as we will dynamically compute symmetry information with it. The following theorem shows that, because of the MC constraints, the longest path in G_p has the same length as the longest path in G_t whenever at least a variable is assigned.

DEFINITION.– *Let $G = (N, E)$ be a graph. Then $maxd(G)$ denotes the size of the longest simple path between two nodes $a, b \in N$.*

THEOREM.– *Suppose we have a subgraph monomorphism instance (G_p, G_t), its associated CSP P, a state S in the search, and suppose the MC_l constraints are arc-consistent. Then if $\exists\ i \in N_p$ such that $|D(x_i)| = 1$, then $maxd(G_p) = maxd(G_t^*)$.*

This is a nice result for complexity issues, when $maxd(G_p)$ is small. In Figure 1.4, $maxd(G_p)$=2 and only nodes at shortest distance 2 from the image of the node 1 in the target graph are included in G_t^*.

The dynamic target graph G can be computed dynamically. In [DEV 05], we showed how subgraph matching can be modelled and implemented in CP(Graph), an extension of CP with graph domain variables [DOO 05]. The domain of a graph variable is modelled by a lower bound and an upper bound graph, and represents all the graphs between the lower and upperbound. 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 $O(1)$.

1.6.1.2. *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) Get G_t^*.

2) The NAUTY and SchreierSims algorithms are called. This returns the new $U_i^{'}$ sets.

3) The main problem is how to adapt the variable and value selection 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 $U_{k-1}^{\prime}$ 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.

1.6.2. *Partial dynamic graphs*

The second techhnique to detect local value symmetries considers the subgraphs associated with the current state.

1.6.2.1. *Detection*

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.

DEFINITION.– *Let S be a state in the search whose variables $x_1, \ldots, x_k$ are assigned to $v_1, \ldots, v_k$ respectively, and $x_{k+1}, \ldots, x_n$ are not assigned yet.*
*The **partial dynamic pattern graph** $G_p^- = (N_p^-, E_p^-)$ is a subgraph of G_p such that :*

- $N_p^- = \{i \in [k+1, n]\}$
- $E_p^- = \{(i,j) \in E_p \mid i \in N_p^- \wedge j \in N_p^-\}$

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

- $N_t^- = \cup_{i \in [k+1,n]} D(x_i)$
- $E_t^- = \{(a,b) \in E_t \mid a \in N_t^- \wedge b \in N_t^-\}$

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 matching problem.

THEOREM.– *Let (G_p, G_t) be a subgraph monomorphism instance, P its associated CSP, and S a state of P during the search, where the assigned variables are $x_1, \ldots, x_k$ with values $v_1, \ldots, v_k$. Let P' be a new CSP of a subgraph monomorphism instance (G_p^-, G_t^-) with additional constraints $x_{k+1}^{'} = D(x_{k+1}), \ldots, x_n^{'} = D(x_n)$. Then:*

1) Each $\sigma \in Aut(G_t^-)$ is a value symmetry of $P^{'}$.

2) Assuming we have the Forward Checking (FC) property, we have $((x_1, v_1), \ldots, (x_n, v_n)) \in Sol(S)$ iff $((x_{k+1}, v_{k+1}), \ldots, (x_n, v_n)) \in Sol(P^{'})$.

Proof sketch When forward checking (FC) is used during the search, in any state in the search tree, every constraint involving *one* uninstantiated variable is arc consistent. In other words, every value in the domain of an uninstantiated variable is consistent with the partial solution. This FC property on a binary CSP ensures that one can focus on the uninstantiated variables and their associated constraints without losing or creating solutions to the initial problem. Such a property also holds when the search achieves stronger consistency in the search tree (Partial Look Ahead, Maintaining Arc Consistency, ...). ■

The computation of G_t^- can be easily performed thanks to graph variables. If T is the graph variable representing the matched target subgraph (with initially $lub(T) = \emptyset$ and $glb(T) = G_t$), then during the computation G_t^- is $lub(T) \setminus glb(T)$.

1.6.2.2. *Breaking*

Breaking local value symmetries is equivalent to breaking value symmetries on the subproblem P'. Puget's method and the dynamic GE-Tree method can thus be applied to the local CSP P'.

1.7. Conclusion

In this work, we present techniques for symmetry breaking in subgraph matching. Specific detection techniques are developed for the variable symmetries and value symmetries. We show that global variable symmetries and value symmetries can be detected by computing the set of automorphisms on the pattern graph and on the target graph and how they can be broken. We also show that local value symmetries can be detected by computing the set of automorphisms on various subgraphs of the target graph. The GE-Tree method is extended to break these local symmetries. Experimental results analyzes the enhancement achieved by global variable and value symmetries. It shows that symmetry breaking is an effective way to increase the number of tractable instances of the graph matching problem.

Ongoing work studies more specifically local variable as well as local value symmetries. Specific techniques are developed for this case [ZAM 07]. Another interesting research direction is the automatic detection of symmetries in graph domain variable.

1.8. Bibliography

[BEL 05] BELDICEANU N., FLENER P., LORCA X., "The tree constraint", *Proceedings of CP-AI-OR'05*, vol. 3524 of *LNCS*, Springer-Verlag, p. 64–78, 2005.

[BEN 94] BENHAMOU B., "Study of symmetry in constraint satisfaction", *PCP'94: Second International Workshop on Principles and Practice of Constraint Programming*, 1994.

[BEN 04] BENHAMOU B., "Symmetry in Not-equals Binary Constraint Networks", *Proceedings of the satelite workshop of CP 2004, Symmetry in Constraints (SymCon'04)*, p. 2–8, september 2004.

[BEN 06] BENHAMOU B., SAÏDI M. R., "Reasoning by dominance in Not-Equals binary constraint networks", *Proceedings of the Twelfth International Conference on Principles and Practice of Constraint Programming (CP-2006)*, LNCS, Cité des Congrès - Nantes, France, Springer, p. 670–674, septembre 2006.

[CAM 04] CAMBAZARD H., BOURREAU E., "Conception d'une constrainte globale de chemin", *10e Journ. nat. sur la résolution de problèmes NP-complets (JNPC'04)*, p. 107–121, 2004.

[COH 06] COHEN D., JEAVONS P., JEFFERSON C., PETRIE K. E., SMITH B. M., "Symmetry Defi nitions for Constraint Satisfaction Problems", *Constraints*, vol. 11, num. 2-3, p. 115–137, Kluwer Academic Publishers, 2006.

[CON 04] CONTE D., FOGGIA P., SANSONE C., VENTO M., "Thirty Years Of Graph Matching In Pattern Recognition.", *IJPRAI*, vol. 18, num. 3, p. 265-298, 2004.

[CRA 96] CRAWFORD J., GINSBERG M. L., LUCK E., ROY A., "Symmetry-Breaking Predicates for Search Problems", AIELLO L. C., DOYLE J., SHAPIRO S., Eds., *KR'96: Principles of Knowledge Representation and Reasoning*, p. 148–159, Morgan Kaufmann, San Francisco, California, 1996.

[DEV 05] DEVILLE Y., DOOMS G., ZAMPELLI S., DUPONT P., "CP(Graph+Map) for Approximate Graph Matching", *1st International Workshop on Constraint Programming Beyond Finite Integer Domains, CP2005*, p. 33–47, 2005.

[DOO 05] DOOMS G., DEVILLE Y., DUPONT P., "CP(Graph): Introducing a graph computation domain in constraint programming", *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, LNCS, Sitges, Barcelona, Spain, Springer, p. 211–215, 2005.

[GEN 01a] GENT I., "A Symmetry breaking constraint for indistinguishable values", *Proceedings of CP'01, SymCon'01 Workshop*, 2001.

[GEN 01b] GENT I., SMITH B., "Symmetry breaking during search in constraint programming", *Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming (CP-2001)*, LNCS, Springer, p. 599-603, 2001.

[GEN 03] GENT I., HARVEY W., KELSEY T., "Generic SBDD using computational group theory", *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP-2003)*, Springer, LNCS, p. 333-346, 2003.

[GEN 05a] GENT I. P., KELSEY T., LINTON S., RONEY-DOUGAL C., "Symmetry and Consistency", *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, vol. 3709 of *LNCS*, Springer, p. 271-285, 2005.

[GEN 05b] GENT I. P., KELSEY T., LINTON S. A., MCDONALD I., MIGUEL I., SMITH B. M., "Conditional Symmetry Breaking", *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, vol. 3709 of *LNCS*, Springer, p. 256-270, 2005.

[LAR 02] LARROSA J., VALIENTE G., "Constraint satisfaction algorithms for graph pattern matching", *Mathematical. Structures in Comp. Sci.*, vol. 12, num. 4, p. 403–422, Cambridge University Press, 2002.

[MAM 04] MAMOULIS N., STERGIOU K., "Constraint Satisfaction in Semi-structured Data Graphs.", WALLACE M., Ed., *CP2004*, vol. 3258 of *Lecture Notes in Computer Science*, Springer, p. 393-407, 2004.

[MCK 81] MCKAY B. D., "Pratical graph isomorphism", *Congressum Numerantium*, vol. 30, p. 35–87, 1981.

[MES 01] MESEGUER P., TORRAS C., "Exploiting symmetries within the constraint satisfaction search", *Artificial intelligence*, vol. 129(1-2), p. 133–163, 2001.

[PUG 05a] PUGET J.-F., "Elimination des symétries dans les problèmes injectifs", *Proceedings des Journées Francophones de la Programmation par Contraintes*, p. 259–266, 2005.

[PUG 05b] PUGET J.-F., "Automatic detection of variable and value symmetries", *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, vol. 3709 of *LNCS*, Springer, p. 477-489, 2005.

[RON 04] RONAY-DOUGAL C., GENT I., KELSEY T., LINTON S., "Tractable symmetry breaking in using restricted search trees", *ECAI'04*, 2004.

[RUD 98] RUDOLF M., "Utilizing Constraint Satisfaction Techniques for Effi cient Graph Pattern Matching.", EHRIG H., ENGELS G., KREOWSKI H.-J., ROZENBERG G., Eds., *TAGT*, vol. 1764 of *Lecture Notes in Computer Science*, Springer, p. 238-251, 1998.

[SEL 03] SELLMAN M., "Cost-based fi ltering for shorter path constraints", *Proc. of the 9th International Conference on Principles and Pratice of Constraint Programming (CP).*, vol. LNCS 2833, Springer-Verlag, p. 694–708, 2003.

[SMI 01] SMITH B., "Reducing symmetry in a combinatorial design problem", *Proc. CP-AI-OR'01, 3rd Int. Workshop on Integration of AI and OR Techniques in CP*, p. 351–359, 2001.

[SOR 04] SORLIN S., SOLNON C., "A Global Constraint for Graph Isomorphism Problems.", RÉGIN J.-C., RUEHER M., Eds., *CPAIOR*, vol. 3011 of *Lecture Notes in Computer Science*, Springer, p. 287-302, 2004.

[SOR 06] SORLIN S., SOLNON C., "A New Filtering Algorithm for the graph isomorphism problem", *Third International Workshop on Constraint Propagation and Implementation*, p. 92–107, 2006.

[ZAM 05] ZAMPELLI S., DEVILLE Y., DUPONT P., "Approximate Constrained Subgraph Matching", *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005)*, LNCS, Sitges, Barcelona, Spain, Springer, p. 832–836, 2005.

[ZAM 07] ZAMPELLI S., DEVILLE Y., SEDI M., BENHAMOU B., DUPONT P., "Breaking Local Symmetries in Subgraph Pattern Matching", *to appear in International Symmetry Conference, Edinburgh,*, 2007.