

Breaking Weak Symmetries¹

Roland Martin

Darmstadt University of Technology
Algorithms Group
64283 Darmstadt, Germany,
martin@algo.informatik.tu-darmstadt.de

Karsten Weihe

The University of Newcastle
School of Electrical Engineering and Computer Science
Callaghan, NSW 2308 Australia
weihe@cs.newcastle.edu.au

Abstract

In this paper we consider a generalisation of symmetry, which we call *weak symmetry*. A weak symmetry acts only on a subset of the variables and preserves the feasibility state only with respect to a subset of the constraints.

We will consider *weakly decomposable problems* that is, a problem decomposes in two subproblems whereby the whole problem contains weak symmetries. That means that one of the subproblems is proper symmetrical.

We discuss a remodelling concept where we use additional variables which we call *SymVar (Symmetry Variable)*. These variables enable us to exploit weak symmetries and achieve symmetry breaking on the symmetrical variables of the problem without losing solutions.

Roughly speaking we rearrange the search tree in a way such that all symmetrical variable assignments are arranged under a specific node. This node represents a complete SymVar assignment of the symmetry class. Therefore the symmetrical variable assignments form a “frontier line” in the search tree where each node corresponds exactly to one symmetrical variable assignment of the SymVars.

1 Introduction

Symmetries transform a (partial) solution in a symmetrical (partial) solution and preserve the state of feasibility: no-goods are transformed in symmetrical no-goods while feasible solutions are transformed in symmetrical feasible solutions. Therefore symmetries decompose the search space in classes of symmetrical solutions whereby each class contains either only feasible or infeasible solutions.

When searching a solution to a problem it is sufficient to find only one solution in each class of solutions. If more than one solution is requested the symmetrical equivalents to the found solution can be derived by applying the symmetry function exhaustively. Therefore symmetries should be excluded from the search space to speed up the search.

Various techniques have been proposed for symmetry handling. In general it is done by remodelling the model, excluding the symmetry up-front via constraints, breaking it during the search or by a combination thereof.

Consider now a scenario where the problem itself includes some kind of symmetry that acts only on a subset of the variables and only with respect to a subset of the constraints. They cannot be excluded or broken by the standard methods mentioned above without losing solutions. This is due to the fact that symmetrical variable assignments for the variables affected by this kind of symmetry yield different value assignments for the asymmetrical variables, such that the crucial property of symmetry – full equivalence of symmetrical solutions – is lost.

A good idea would be to decompose the problem in two parts and perform symmetry breaking on the symmetrical subproblem. This implies to solve them successively and use the solutions of the first model as input to the second one. But solving the first model exhaustively is usually too expensive.

SymVars address this problem. SymVars are additional variables which represent the symmetrical solutions of the problem. The symmetry function is then applied via the assignment of the SymVars. In case the symmetry function is a permutation, for example, an assignment of the SymVars represents a permutation of the variables of the symmetrical subproblem.

Using SymVars seems to be particularly useful for tight-fit problems and optimisation within a given time limit (that is just a small amount or a fraction of the time it would take to search the problem exhaustively).

In Section 2 we introduce weak symmetries and state in Section 3 how weak symmetries influence the search tree. Section 4 comprises some variations of standard problems that include weak symmetries. In Section 5 we discuss the different ways of handling weak symmetries and introduce in Section 6 the new modelling concept of SymVars and how it could be used to break weak symmetries. In Section 7 we give examples that use the SymVar concept and in Section 8 we give an outlook to further work with weak symmetries and SymVars.

¹In cooperation with Philips/Assembléon, Netherlands

2 Weak Symmetry Description

2.1 Weak Symmetry Definition

To characterise weak symmetries we first have to define a weakly decomposable problem.

Definition 1 (Weakly Decomposable Problem)

Given a problem description $P = (X, C)$.

$X = \{x_1, \dots, x_n\}$ is the set of variables and

$C = \{c_1, \dots, c_m\}$ is the set of the constraints. The constraints c_1, \dots, c_k are of the form $c(x_1, \dots, x_\ell)$ and the constraints c_{k+1}, \dots, c_m are of the form $c(x_1, \dots, x_n)$.

P is **weakly decomposable** if it splits in two subproblems P_1 and P_2 .

$P_1 = (X_1, C_1)$ consists of the variables $X_1 = \{x_1, \dots, x_\ell\}$ and the constraints $C_1 = \{c_1, \dots, c_k\}$ and

$P_2 = (X_2, C_2)$ consists of the variables $X_2 = \{x_{\ell+1}, \dots, x_n\}$ and the constraints $C_2 = \{c'_{k+1}, \dots, c'_m\}$ of the form $c(d_1, \dots, d_\ell, x_{\ell+1}, \dots, x_n)$, whereby d_i is a concrete value for the variable x_i , $1 \leq i \leq \ell$.

In that, for $\ell + 1 \leq j \leq m$, c'_j arises from c_j by assigning the variables of X_1 concrete values: $x_i = d_i$, $1 \leq i \leq \ell$.

If $\ell = 0$ then P_1 is empty and if $\ell = n$ then P_2 is empty.

Clearly, if $\ell < n$ then the problem is a **proper weakly decomposable problem**.

A symmetry that acts on the subproblem P_1 is considered a weak symmetry.

Definition 2 (Weak Symmetry)

Given a weakly decomposable problem with subproblems P_1 and P_2 . A symmetry acting on P_1 but not on P_2 is called a **weak symmetry**.

If the weakly decomposable problem is proper then the symmetry is considered a **proper weak symmetry**.

If the weakly decomposable problem is not proper then the symmetry is a real symmetry.

In the following we will only consider proper weakly decomposable problems and proper weak symmetries for convenience.

A weak symmetry splits the problem into two parts: one that is symmetrical and one that is asymmetrical.

The symmetrical part contains all the constraints and variables that are affected by the weak symmetry.

The asymmetrical part contains the rest of the constraints and variables.

Both parts cannot be solved individually to receive a full solution.

Side remark: In our studies so far it is useful to regard the symmetrical part first and then regard the asymmetrical part. This is since the asymmetrical part in our studies is mostly concerned with deriving the objective value (in case of optimisation) or with constraints that evaluates the objective value (in satisfaction problems where a solution must at least achieve a certain objective value to be feasible). \square

2.2 Weak Symmetry as a Constraint Pattern

To characterize the weak symmetry problem we state it following the definition of constraint patterns ([Walsh, 2003]).

- Pattern name: WeakSymmetry
- Context: A symmetry that acts only on a subset of the variables respecting only a subset of the constraints in terms of feasibility. The symmetry splits the problem in a symmetrical and an asymmetrical part intimately related to each other.
- Problem: Breaking the weak symmetry without losing solutions.
- Forces: Breaking the weak symmetry is not possible without losing solutions and not breaking it is highly inefficient.
- Solution: Using SymVars results in a rearrangement of the search-tree that allows all known kinds of standard symmetry breaking for weak symmetries.
- Example: Weighted n-Queens, Multiple Rack Configuration, Automated Manufacturing

3 The Effect of Symmetries in the Search Tree

For convenience, we consider a static variable ordering (where concerning weak symmetries the symmetrical variables are consecutively ordered starting at the root of the tree) but the ideas can also be applied to a dynamical variable ordering.

3.1 Proper Symmetries in the Search Tree

If a problem contains symmetries all paths of an equivalence class of solutions of (partial) solutions can be identified and represented by a single path. Therefore the pruned search tree contains only a fraction of the paths of the straight forward search tree.

In practice, this means that symmetrical (partial) solutions are evaluated as no-goods as soon as possible and the underlying subtree is pruned.

Therefore the explored search tree also excludes all but one path per equivalence class if the symmetry is broken.

3.2 Weak Symmetries in the Search Tree

If a problem contains weak symmetries the symmetry function can only be applied to nodes up to a specific depth in the search tree since up to this depth the symmetric variables of P_1 are arrayed. Therefore only paths up to this specific level could be identified. But the subtrees under each symmetrical subpath of a class are different. Therefore these paths cannot be identified and represented by a single path without losing solutions in the subtree. These solutions **cannot** be derived using the weak symmetry function.

Therefore the search tree contains all symmetrical paths.

To our best knowledge there is no way of excluding weak symmetries without losing solutions in this search tree.

4 Weak Symmetries in Practice

Problems including weak symmetries can be found in optimisation as well as in satisfaction problems. Also several standard problems can be extended to fit in the class of proper weakly decomposable problems. The easiest way to extend a problem to be proper weakly decomposable is to cluster two problems that are familiar in some way like the magic-knight-tour which is a combination of a knight tour and a magic square [Jelliss, 2004], [Stertenbrink, 2004].

4.1 Proper Weakly Decomposable Optimisation Problems

Every optimisation problem can be considered weakly decomposable as follows: The objective value can be represented by a variable and the objective function is a constraint that determines the value of the objective variable. Therefore the original problem forms P_1 and the optimisation part forms P_2 of the weakly decomposable problem.

If the objective function is in the same way symmetrical as the weak symmetry in P_1 then the problem is not proper weakly decomposable.

There are optimisation problems where P_2 consists of further variables and constraints. In such a case the variable assignment of these further variables depend on the variable assignment of P_1 .

An example of such a problem is the “open warehouse” problem where the number of sold goods is to be maximised (See [Hentenryck, Lustig, 1999] for a detailed description). There are several stores and each can only sell goods if they are open. The variables of P_1 would be the decision variables whether a store is open or not while the variables that state the number of goods sold are a part of P_2 .

This is demonstrated by a short model using ILOG OPL Syntax [Hentenryck, Lustig, 1999]:

```
int+ numberOfStores;
range Stores 1..numberOfStores;

int+ numberOfGoods;
range Goods 1..numberOfGoods;

int+ InStock[Stores,Goods] =...;

var Open[Stores] in 0..1;
var Sold[Stores,Goods] in 0..maxint;

...

forall (s in Stores)
Open[s]= 0 => Sold[s, g in Goods] = 0;
```

4.2 Proper Weakly Decomposable Satisfaction Problems

Satisfaction problems are often self-contained such that they cannot be decomposed reasonably. Nonetheless many can be extended to be proper weakly decomposable. In the case of extension the original problem forms P_1 and the extension

forms P_2 . The variable assignment of the variables in P_1 predefines the feasible domains for the variables in P_2 .

If the problem is weakly symmetrical this could lead to the fact that not all partial symmetrical solutions of a class are feasible with respect to the problem P . So the feasibility state for P_1 is preserved but not in general for P . That means that P_2 contains constraints that evaluates a partial solution infeasible.

Consider a problem where depending on a variable assignment for the symmetrical part variables are to be assigned. The variable assignment of the symmetrical part therefore predefines feasible variable assignments for the latter variables whereby even symmetrical solutions (for the symmetrical part) may yield different feasible variable assignments for the asymmetrical part.

This may even have the consequence that a symmetrical variable assignment yields only infeasible solutions.

4.3 Weak Symmetries in Extensions of Standard Problems

Example of an Optimisation Problem – Weighted N-Queens

Consider the *weighted n-Queen problem* where each field on the chessboard features a weight expressed by an integer. The task now is to find a n-Queen solution with minimal/maximal weight.²

The symmetrical part P_1 is clearly the n-Queen problem while the asymmetrical part P_2 is determining the objective value.

The weak symmetries acting on the variables of P_1 are all kinds of rotation and flips of the chessboard.

In this problem for example the flip around the x-axis delivers a symmetrical solution but this may have a different objective value and is therefore a different solution to the problem.

Example of a Satisfaction Problem – Time-based Rack Configuration

Consider the *time-based rack configuration problem*, a variation of the rack configuration problem (CSPLib problem031 [Gent, Walsh, 2003], [Kiziltan, Hnich, 2003]).

In the rack configuration problem a number of racks of different types are needed to plug in a certain number of different electronic cards. The task is to decide how many racks of each type are needed such that the price for all racks is minimal.

In the extension of the problem the price is modelled as an upper bound, that reflects the budget that could be spend at most for all racks. Any rack configuration that's price is below the budget is a feasible configuration. But additionally to the decision which racks are to be bought the time of purchase has to be considered or more specifically the order in which they are bought. Also for some kind of electronic cards there are constraints that state when they have to be used in the latest.

The symmetrical part P_1 is the rack configuration (configured racks are feasible no matter in which order they appear)

²This problem could also be stated as satisfaction problem when a n-queen solution is wanted that achieves at least a specified value.

while the order of the racks is the asymmetrical part P_2 (since the order defines the feasibility of the racks with respect to the time constraints of the electronic cards).

The weak symmetry acting is the permutation of the racks. Any permutation of the racks is still a feasible rack configuration. But a specific order of the racks may be infeasible since a constraint stating the latest use of a specific card is violated.

Example from Automated Manufacturing

The problem formulation is a variation of a real world problem originating from a cooperation with Philips/Assembléon, Netherlands (see Fotenote 1). For a more specific descriptions see [Gaudlitz, 2004] and [Tazari, 2003].

Given is:

- A mounting machine with several in parallel working, consecutively ordered mounting devices.
- Several tools and container of a specific type for specific component types.
- A PC board layout specified by a set of mounting tasks, each consisting of a specific component type and coordinates where this component type has to be placed on the board.

The task is to find a setup for each mounting device – consisting of a tool and several component container – and – depending on this setup – find an optimal distribution of the mounting tasks to the mounting devices such that the workload is minimised.

The mounting devices are consecutively ordered and do not interfere with each other. The boards are arrayed along the mounting devices. Usually a board is visible on two mounting devices at a time such that both can mount on it, each on a different clipping of the board.

The mounting devices are identical in terms of the possibilities of assigning a setup but each has a different visibility range on the machine. Therefore each device has access to a different subset of mounting tasks of the boards.

Each mounting device receives its own setup that defines which mounting tasks can be processed by this device. Since the devices are symmetrical feasible setups can be swapped to other devices and they are still feasible. A specific setup achieves different performance values in terms of the job processing since the subsets of accessible mounting tasks are different for each mounting device.

The symmetrical part P_1 of the problem is the configuration of the setups for the individual mounting devices while the asymmetrical part P_2 is the distribution of the mounting tasks to the devices.

The weak symmetry acting is the permutation of the setups for the mounting devices since a feasible setup is feasible no matter to which mounting device it is assigned.

5 Handling Weak Symmetries

For the purpose of this paper we focus on the case that the weak symmetry function acts like a permutation.

5.1 Standard Methods

Usually a problem that contains weak symmetries could be decomposed in two independent subproblems whereby the

solution of the first is input to the second. The advantage of this method is that the weak symmetry in this case is a real symmetry in the subproblem and can be broken. All symmetrical solutions can be derived explicitly in post-processing to form the input for the second subproblem. The drawback is that the first problem has to be solved exhaustively to get all solutions before the second problem can be solved. But if one is just interested in one solution that satisfies all constraints (CSP) or the best solution that can be found in a specified time (CSOP) this approach would not be suiting.

A different approach would be to have a framework that supports several models and anytime a solution for the first problem is found it is passed to the second model to find a solution for it like ILOG OPLScript ([Hentenryck, Lustig, 1999]).

Both approaches would lead to a remodelling of the problem and either a special framework has to be used or the solving process is split since the solution of one problem has to be post processed to form the new input.

The approach we suggest to handle weakly decomposable problems is using variable objects called *SymVar*. Using *SymVars* enables us to use standard symmetry breaking methods without solving the first problem exhaustively and without the need of a special framework.

This approach also means to remodel the problem but only in the way of introducing new variables.

5.2 Breaking Weak Symmetries

The main problem in breaking weak symmetries is that the knowledge about the symmetrical structure of P_1 can not be exploited without losing solutions.

Since we are interested in symmetrical equivalents of a feasible solution to P_1 we do not care for infeasible solutions. An infeasible solution to P_1 is also infeasible to P_2 such that the state of infeasibility is preserved throughout the search. Therefore all infeasible classes can be excluded without doubt.

Moreover we are only interested in feasible solutions to P_1 not in feasible partial solutions. Therefore it would be sufficient to retrieve all symmetrical solutions of an investigated feasible solution and exclude the paths to that solutions. That means that we basically have to find only one representative of a feasibility class. That is exactly what symmetry breaking does.

So if we can retrieve all symmetrical solutions we can use all known kinds of symmetry breaking.

To retrieve all symmetrical solutions we use *SymVars* that state which symmetrical equivalent of a solution we consider further for P_2 .

6 The SymVar Concept

The main problem in handling weak symmetries is to receive all symmetrical solutions (once a solution is found) without searching in the symmetrical branches of the search tree.

To achieve this we introduce additional variables that act as an interface to the symmetry of the problem. These variables are the *SymVars*.

Formally the original problem description P is changed to P' which splits in the subproblems P'_1 , P_{sym} and P'_2 .

P_1 is extended to P'_1 by adding the symmetry breaking constraints. P_{sym} contains the SymVars and the constraints for the SymVar assignment. In fact the weak symmetry function is expressed by the constraints of P_{sym} .

P_2 is changed to P'_2 in the way that instead of the values for the variables of P'_1 the symmetrical equivalents expressed by the variables of P_{sym} are considered.

$P_{sym} = (X_{sym}, C_{sym})$ consists of the variables $X_{sym} = \{x_{s_1}, \dots, x_{s_r}\}$ and the constraints $C_{sym} = \{c_{s_1}, \dots, c_{s_t}\}$ of the form $c(x_{s_1}, \dots, x_{s_k})$.

P'_2 consists of the variables $X_2 = \{x_{\ell+1}, \dots, x_n\}$ and the constraints $C'_2 = \{c''_{k+1}, \dots, c''_m\}$, of the form $c(Sym(d_1), \dots, Sym(d_\ell), x_{\ell+1}, \dots, x_n)$.

$Sym(d_1), \dots, Sym(d_\ell)$ is a symmetrical solution – determined by the value assignment of the SymVars – of the concrete values for the variables x_1, \dots, x_ℓ .

That means that the difference between c'_j and c''_j is that the concrete values of the variables of X_1 are changed to a symmetrical value assignment. Which symmetrical values assignment that is, is determined by the values of the variables of X_{sym} .

Definition 3 (Symmetry Variable (SymVar))

The variables of X_{sym} are the **SymVars**. Each SymVar is a copy of a set of variables of the symmetrical problem P_1 . The set of SymVars represents all variables of the problem P_1 .

A value assignment of the SymVars represents a symmetrical solution for the variables of the problem P_1 .

Therefore the symmetry of the problem acts on the SymVars rather than on the initial variables X_1 such that the symmetry on the initial variables is broken.

An instance of a SymVar corresponds to a set of variables of P_1 forming a specific object in the model.

In the weighted n -Queens problem this object would be a single queen while in the time-based rack configuration problem it would be a whole rack and its included electronic cards. In the last context a value assignment to a SymVar that represents a specific rack would assign this rack to a specific time slot.

In case of the example from automated manufacturing from Page 4 a mounting device would be the corresponding object. A specific SymVar would in this case correspond to a specific setup for a mounting device while the value of the SymVar indicates to which specific mounting device this setup is assigned.

A permutation of the SymVar represents now a permutation of the variable assignments for the mounting devices. Therefore the symmetry of the mounting devices – the permutation of the devices – can be broken since all symmetrical solutions can be derived by the instantiations of the SymVars. The symmetry acts now only via the SymVars and is broken on the initial variables V_1 .

6.1 Ordering Heuristics for P_{sym}

The gain in using SymVars is that once a feasible solution to P'_1 is found all symmetrical solutions can be found just by solving P_{sym} and therefore the symmetry in P'_1 can be broken.

When the SymVars assignment is infeasible for P'_2 or another solution is to be evaluated, a different solution of P_{sym} is considered.

Basically this means that each solution of P_{sym} has to be considered since it may yield a different solution for P'_2 .

But depending on the problem and scenario not all symmetrical solutions have to be considered. Propagation can still take place and therefore ordering heuristics can be used to guide the search. In the automated manufacturing problem (Page 4) it would be a good idea to assign a setup to a mounting device such that as many mounting tasks as possible are visible.

In case of optimisation a good solution may rule out many other such that a great deal of symmetrical solutions can be pruned very early.

Ordering heuristics depend very much on the problem itself and finding a good one is still a crucial task.

But to our best knowledge there are no limitations in terms of using ordering heuristics.

6.2 Trade off in Using SymVars

Since P' contains more variables than the original problem P it means basically that it takes more variable assignments. In fact the height of the tree to make up a solution is greater than the height of the tree in P .

On the other hand, since the weak symmetry is broken in P' the width of its search tree is lesser than the width of P .

Depending on the problem, the instance, the symmetry and the scenario, SymVars may or may not lead to speed up in the search for better solutions.

Disadvantage Scenario

Consider a CSP where the first found solution of P'_1 does not lead to a feasible solution of P' and propagation is unable to prune efficiently. That would mean that all symmetrical solutions of P'_1 will be considered without finding a feasible solution.

The standard approach does not consider the symmetrical solutions necessarily and therefore it could happen that the it finds a feasible solution while the SymVar approach still considers the symmetrical solutions.

Advantage Scenario

Consider an optimisation problem with a given computational time limit and consider further that the solution space is rather sparse but propagation can not prune effectively. Consider further that it is rather difficult to find a solution to P_1 but very easy to find a solution to P_2 .

The standard approach would invest much time in searching a feasible solution to P_1 potentially wasting much time with the unbroken weak symmetry. Even if a solution is found it could take a long time until the next is found because of the sparse solution space. Therefore it is likely that this approach does not find many solutions within the given time limit.

In the SymVar approach – once a solution for P'_1 is found – the whole equivalence class to this solution is at hand via the SymVars. Since it is easy to solve P'_2 many solutions are at hand. Therefore this approach is more likely to find many solutions within the given time limit increasing the chance of finding a good solution.

7 Examples using SymVars

In this section we provide insight to the usage and construction of SymVars for some problem descriptions.

7.1 Weighted n-Queen Problem with SymVars

In the weighted n-queen problem a SymVar represents a single queen. Since a geometrical symmetry is more complicated in comparison to a permutation, the SymVars act in a different way: A value assignment for a specific SymVar determines which symmetrical value its corresponding queen has (suppose each queen denotes a column) as well as its column on the chessboard. That means that the row value as well as the column representation are subject to the symmetry.

Example:

Consider the diagonal flip the symmetry function. That means that a queen on column i with row value j is translated to column j and row i .

The problem P' consists of the following parts:

P'_1 is just the n-queen problem extended by the symmetry breaking constraints for the diagonal flip.

P_{sym} consists of determining the symmetrical solutions (which in this case is the identity and the diagonal flip).

P'_2 consists of the problem for determining the objective value whereby the symmetrical values determined by P_{sym} are considered (instead of the values determined by P'_1).

P_{sym} is realised by

$$X_{sym} = \{q_{1_{sym}}, \dots, q_{n_{sym}}\} \text{ and}$$

$$C_{sym} = \{(q_{i_{sym}}=q_i), 1 \leq i \leq n \vee (q_{i_{sym}} = j | q_j = i), 1 \leq i \leq n\}.$$

Since the diagonal flip contains only two different values the constraint in C_{sym} just states these two symmetrical values as feasible.

It's also possible to break the other symmetries of the problem like rotations and any combination thereof. They just have to be incorporated in C_{sym} .

7.2 Automated Manufacturing – Revisited

We consider again the automated manufacturing problem from Page 4 and investigate how SymVars could be used to break the weak symmetry.

P'_1 still consists of the setup configuration part such that the task is to find a feasible setup for the mounting machine. In addition the symmetry of the permutation of the setups is broken.

Each setup for a mounting device is represented by a SymVar, such that its position on the machine is not determined yet. The task for P_{sym} therefore is to find a permutation of the setup delivered by P'_1 on the machine. This can be easily done by stating an alldifferent constraint on the SymVars.

The task of P'_2 is to find a mounting task distribution based on the permuted setup delivered by P_{sym} and compute the processing time for this distribution.

When an optimal distribution for this setup is determined a different permutation is considered for P'_2 .

7.3 Variation of the Problem

Following we will consider a variation of the above problem. The subproblem P'_2 itself is hard to solve and finding an optimal distribution comprises a lot of work. There exists an

algorithm that solves the problem by using a network flow as core algorithm but the runtime is also exponential.³ But since we want to investigate the usage of SymVars we focus on solving P_1 and choose a heuristic for P_2 .

Our studies on the problem indicate that good solutions for P_2 can be found for a setup with a high visibility of the mounting tasks. Therefore the problem P is to find a feasible machine setup and maximise the number of visible mounting tasks. That means that the task for P_2 is just to compute the number of visible mounting tasks for the setup.

Given are:

- a variable matrix $A^{m \times n}$ (representing the mounting machine, whereby n represents the number of mounting devices and m represents the number of component container that can be assigned to each device)
- a number t of different types (representing the component types)
- $n \times m$ values $d_1, \dots, d_{n \times m} \in [1, \dots, t]$ that have to be assigned to A (representing the component container)
- a matrix $V^{n \times t}$ (representing the number of mounting tasks for each component type that are processable on each mounting device)
- several sets containing values in $[1, \dots, t]$ (representing the tool types)

Objective: Find a distribution of the values $v_1, \dots, v_{n \times m}$ to the matrix A such that the number of processable mounting tasks is maximised:

$$\max \sum_{(col \in 1..n, row \in 1..m)} V[col, A[row, col]]$$

Constraints:

- several types cannot be assigned simultaneously. This means that all values in a column must be in the same compatibility set
- the values in each column must be all different

Since the order in which the values are assigned in a column does not matter the values can be ordered increasingly breaking the row symmetry.

Using SymVars

P_1 is extended to P'_1 by adding the column breaking constraints.

P_{sym} consists of

$$X_{sym} = \{pos_1, \dots, pos_n\} \in \{1, \dots, n\} \text{ and}$$

$$C_{sym} = \{alldifferent(pos_1, \dots, pos_n)\}.$$

Each SymVar pos_i , $1 \leq i \leq n$ represents its corresponding column i and therefore the values assigned to this specific column in P'_1 . A value assignment $pos_i = j$ then yields, that the values of column i are permuted to column j .

P'_2 determines the objective value and consists of

$$X_2 = \{obj\} \text{ and}$$

$$C_2 = \{obj = \sum_{(col \in 1..n, row \in 1..m)} V[col, A[pos_{col}, row]]\}$$

³See [Gaudlitz, 2004] and [Tazari, 2003] for an implementation of the algorithm.

Preliminary Results

We generated several small instances of the problem to investigate the power of using SymVars. We used matrices of several sizes ranging from 5×6 to 5×10 and different number of types for the values. We also set a time limit of 20 minutes for the search and compared the results of the naive model with the one that uses SymVars. We used ILOG OPL Studio 3.7 to do the computation on a Pentium 4 with 3.2 GHz and 512 MB RAM.

The results (although preliminary) were very encouraging. In all instances the SymVars approach did find the first solution in less time compared to the naive approach. Even if the objective value was lesser for the SymVars approach on this first solution, another solution could be found (in the same or lesser time that it took the naive approach to compute the first solution) with a objective value that was better (sometimes up to 20 %) than the solution of the naive approach. Even more encouraging was that for all instances the objective value of the SymVar approach was even or much much better than the objective value of the naive approach and mostly the computational time was just a fraction of the solving time for the naive approach.

For example in an instance where A has the dimensions 5×10 the naive approach did find a first solution with objective value 230 after 256.7 seconds and found the best solution with objective value 257 after 580.2 seconds. The SymVar approach found its first solution with objective value 233 after 0.218 seconds, found its seventh solution with objective value 265 after 2.812 seconds and its best solution with objective value 276 after 305 seconds.

This small example shows that the SymVar approach seems to be very promising and that it is especially useful for time bounded optimisation.

8 Outlook

We introduced the weakly decomposable problem definition, the weak symmetry definition and delivered some extensions of well examined standard problems that contain weak symmetries.

We also introduced SymVars that represent symmetrical solutions of a subproblem with the consequence that the symmetry can be broken on the subproblem.

In addition we presented some problems where SymVars are applied and delivered preliminary computational results for on problem that were very encouraging.

An open question is whether there are other scenarios, where weak symmetries play a vital role and whether they can also be handled there in the way we introduced.

If the weak symmetry does not preserve the state of feasibility for the whole problem then there should be ordering heuristics such that not all $n!$ permutations have to be considered (in the case the symmetry is a permutation). Developing such heuristics would also be very interesting.

References

[Apt, 2003] Krzysztof Apt *Principles of Constraint Programming* Cambridge University Press, 2003

[Backofen, Will, 1999] R. Backofen and S. Will *Excluding Symmetries in Constraint-Based Search* In: Principles and Practice of Constraint Programming, pp. 73-87, 1999

[Flener et al., 2002] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, T. Walsh *Matrix Modelling: Exploiting Common Patterns in Constraint Programming* In: Proceedings of the International Workshop on Reformulating Constraint Satisfaction Problems, 2002

[Flener et al., 2002] P. Flener, A. Frisch, B. Hnich, Z. Kiziltan, I. Miguel, J. Pearson, T. Walsh *Breaking Row and Column Symmetries in Matrix Models* In: 8th International Conference on Principles and Practices of Constraint Programming (CP-2002), Springer, 2002

[Gaudlitz, 2004] Rico Gaudlitz *Optimization Algorithms for Complex Mounting Machines in PC Board Manufacturing* Diploma Thesis, TU Darmstadt, To appear Oktober 2004

[Gent, Smith, 1999] I. Gent, B. Smith *Symmetry Breaking During Search in Constraint Programming* School of Computing Research Report 99.02, University of Leeds, 1999

[Gent, Smith, 2000] I. Gent, B. Smith *Symmetry Breaking in Constraint Programming* In Horn, W., ed.: Proceedings of the 14th European Conference on Artificial Intelligence, pp. 599-603, 2000

[Gent, Walsh, 2003] I. Gent, T. Walsh, B. Selman *CSPLib: a problem library for constraints* <http://4c.ucc.ie/tw/csplib>

[Hentenryck, Lustig, 1999] V. Hentenryck, I. Lustig *The OPL Optimization Programming Language* The MIT Press

[Jelliss, 2004] Compiled by George Jelliss *Knight's Tour Notes* <http://www.ktm.freeuk.com>

[Kiziltan, Hnich, 2003] Z. Kiziltan, B. Hnich *Prob031: Rack Configuration Problem* <http://4c.ucc.ie/tw/csplib/prob031>

[Puget, 2003] J.-F. Puget *Symmetry Breaking Using Stabilizers* In Rossi, F., ed.: Proceedings of 9th International Conference on Principles and Practice of Constraint Programming (CP2003), Springer, 2003

[Smith, 2000] B. Smith *Modelling a Permutation Problem* In: Proceedings of ECAI 2000 Workshop on Modelling and Solving Problems with Constraints, 2000

[Smith, Gent, 2001] B. Smith, I. Gent *Reducing Symmetry in Matrix Models: SBDS v. Constraints* Held on SymCon'01, 2001

[Sternbrink, 2004] Hosted by Guenter Sternbrink *Computing Magic Knight Tours* <http://magictour.free.fr>

[Tazari, 2003] Siamak Tazari *Solving a core scheduling problem in modern assembly-line balancing* Technical Report, TU Darmstadt, Oktober 2003

[Walsh, 2003] Toby Walsh *Constraint Patterns* In Rossi, F., ed.: Proceedings of 9th International Conference on Principles and Practice of Constraint Programming (CP2003), Springer, 2003