Weak Symmetries in Problem Formulations

Student name: Roland Martin
Supervisor name: Karsten Weihe

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

Abstract. In this article we will present some problems that have weak symmetries. In contrast to a proper 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. Therefore, breaking weak symmetries with standard techniques would lead to a loss of solutions.
This article presents also a modelling technique which is based on additional variables called *SymVar – Symmetry Variable*. These variables enable us to achieve symmetry breaking on the symmetric variables of the problem without losing solutions. By using this technique we are able to break the weak symmetry without losing solutions.
We will show for three problems how SymVars are applied such that the weak symmetries can be broken.

1 Introduction

Symmetries transform (partial) solutions into symmetric (partial) solutions whereby the property of feasibility is not changed: no-goods are tranformed into symmetric no-goods while feasible solutions are transformed into symmetric feasible solutions. Therefore, symmetries decompose the search space into classes of symmetric solutions, whereby each class either contains feasible solutions only or infeasible solutions only.

When searching for all solutions to a problem it is sufficient to find only one solution in each class of solutions. The symmetric equivalents can be derived by applying a symmetry function exhaustively to each class after the search process. Therefore, symmetries should be excluded from the search space to speed up the search.

Weak symmetries act only on a subset of the variables and/or satisfy only a subset of the constraints of the problem. Therefore, weak symmetries preserve the state of feasibility only with respect to the subset of variables they act on and only for the constraints they satisfy. This means if two solutions are symmetric under the weak symmetry they yield different full solutions with potentially different feasibility states.

But weak symmetries cannot be simply broken, since this would result in a loss of solutions that cannot be derived afterwards.

Nonetheless this article presents a modelling technique that enables us to deal with weak symmetries such that they can be broken without losing solutions.

Weak symmetries occur in many fields of applications and have already been discovered and identified in planning, scheduling and model checking ([1] - [4]). Also extensions to classical problems like the rack configuration problem ([5])

Section 2 gives the definitions of weak symmetries and SymVars and present how they are basically used in modelling.

Section 3 introduces three problems that have weak symmetries and show how the SymVars can be applied. In Section 4 we conclude and give an outlook to future work.

1.1 Prerequisites

We characterize a satisfaction problem by $P = (X, C)$, whereby $X = \{x_1, \dots, x_n\}$ is the set of variables and $C = \{c_1, \dots, c_m\}$ is the set of constraints.

For an optimisation problem we just extend this formulation to $P = (X, C, f)$, where X and C are defined as above and f is the objective function.

A solution to P is denoted by
$s_P = (x_1 = v_1, \dots, x_n = v_n) = (X)$. This means that each variable in X is assigned a value of its corresponding domain.

2 Weak Symmetries

2.1 Weak Symmetry Definition

Weak symmetries act on problems with special properties. To characterize weak symmetries we first define weakly decomposable problems. The goal is to find a decomposition such that one of the sub-problems contains all symmetric variables and constraints and the other not.

Definition 1 (Weakly Decomposable Problem)
A problem $P = (X, C)$ is **weakly decomposable** *if it decomposes into two subproblems $P_1 = (X_1, C_1)$ and*
$P_2 = (X_2, C_2)$ with the following properties:

$$X_1 \cap X_2 \neq \emptyset \tag{1}$$

$$X_1 \cup X_2 = X \tag{2}$$

$$C_1 \cup C_2 = C \tag{3}$$

$$C_1 \cap C_2 = \emptyset \tag{4}$$

$$C_2 \neq \emptyset \tag{5}$$

The first property states that P_1 and P_2 contain a subset of shared variables (namely $X_1 \cap X_2$). These variables have to assume the same values in both subproblems to deliver a feasible solution to P. Therefore they link both problems. Without that restriction the problem would be properly decomposable. The second and third property states that none of the variables and constraints of the original problem P are lost. Furthermore the third and fourth property state that C_1 and C_2 is a partition of C. Basically this is not necessary for feasibility. A constraint could be in both subsets (if defined on $X_1 \cap X_2$

only) but would be redundant for one of the problems because the solution to the other subproblem would already satisfy this constraint. Therefore, this is just a question of efficiency. The last property states that P_2 is not allowed to be unconstrained. But note that this restriction does not hold for P_1 since we want P_1 to be the symmetric problem and an unconstrained problem is perfectly symmetric.

An example (besides the weighted n-queens problem) for a weakly decomposable problem is also the magic knight tour. (See [6] and [7]). In this problem a knight tour on a chessboard is sought for where the numbers of the moves constitutes a magic square. The weakly decomposition is that P_1 consists of the magic square problem and P_2 constitutes that when following the numbers 1 to n^2 this is a knight tour.

A symmetry that acts on the subproblem P_1 (but not on P_2) is considered a weak symmetry.

Definition 2 (Weak Symmetry)
Given a weakly decomposable problem P with a decomposition (P_1, P_2).
A symmetry S on P_1 is called a **weak symmetry** *on P with respect to the decomposition $P_1, P_2)$ iff S acts on P_1 but not on P_2.*

The intention of the decomposition of the problem is that X_1 contains all symmetric variables (and only these) and X_2 contains also the rest of the variables.
The gain is that we get a subproblem that is not affected by the weak symmetry (P_2) and a subproblem where the weak symmetry affects all variables and all constraints (P_1).

2.2 Weak Symmetry Breaking

Since the weak symmetry does not act on the whole problem, it cannot be broken on the whole problem. But it can be broken in P_1 (where it acts as a proper symmetry). This means that in the search tree equivalent solutions of P_1 are identified with each other. On the other hand we would lose the symmetric solutions if we broke the symmetry. Therefore we need a way to represent these solutions explicitly because they are needed in order to solve P_2 which delivers a full solution for P.

In order to represent these symmetric solutions we introduce additional variables called *SymVars*. These variables encode the symmetric equivalents of a solution. Also additional constraints are needed that constrain the SymVars to take only values that state a feasible symmetric equivalent of P_1. Therefore a new sub-problem $P_{sym} = (X_{sym}, C_{sym})$ is introduced. X_{sym} are the SymVars and C_{sym} are the constraints that constitutes a feasible symmetric equivalent.

The solving order now is: P_1 — P_{sym} — P_2.

Notation 1 *Let P be a weakly decomposable problem with a decomposition (P_1, P_2), $P_1 = (X_1, C_1), P_2 = (X_2, C_2)$.*
Let $X_{sym} = \{y_1, \dots, y_\ell\}$ be a set of SymVars that constitutes the variables of the subproblem P_{sym}.
A solution to P_1 is denoted by $s_{P_1} = (X_1)$.
A solution to P_{sym} is denoted by $s_{P_{sym}} = (X_1, X_{sym}) = (s_{P_1}, X_{sym})$
A solution to P_2 is denoted by $s_{P_2} = (X_1, X_{sym}, X_2) = (s_{P_{sym}}, X_2)$.

A solution to P_2 is automatically a solution to P.
Let $s_{P_1} = (v_1, \ldots, v_n)$ be a solution to P_1, where v_i is a value of the domain of x_i, $i \in \{1, \ldots, n\}$.
A solution $s_{P_{sym}} = (s_{P_1}, v'_1, \ldots, v'_\ell)$ is a symmetric solution to s_{P_1}, where v'_j is a value of the domain of y_j, $j \in \{1, \ldots, \ell\}$

The solving order in more detail is to search a solution s_{P_1} to P_1, determine a symmetric equivalent $s_{P_{sym}}$ in P_{sym} and use this solution to determine a solution to P_2 which already states a solution to P. Note that when considering P_{sym} the variables X_1 are already assigned. The same holds for P_2 with X_1 and X_{sym}.

Consequences:

- Every feasible value assignment to the SymVars constitutes a symmetric solution to s_{P_1}
- None of the values in X_1 have to be reconsidered to receive a symmetric equivalent
- The symmetry can be broken in P_1 because all symmetric solutions to S_{P_1} are expressed by $s_{P_{sym}}$

Note that there is not necessarily one SymVar for each variable in X_1. Often it holds that $|X_{sym}| < |X_1|$.

To solve P we consider the partial solution $s_{P_{sym}}$. When a solution is found the search backtracks and reconsiders values for the SymVars to determine a new solution. All these solutions are symmetric equivalents to the solution s_{P_1}. Only when the search backtracks and reconsiders variables in X_1 a solution for a different equivalence class can be found.

By using SymVars we can break the symmetry in P_1 but do not lose any symmetric solution in an equivalence class.

3 Problems Containing Weak Symmetries

There are a lot of problems that have weak symmetries. Weak symmetries can be defined on weakly decomposable problems like done here. Therefore standard problems containing symmetries can be extended to have weak symmetries. One possibility to do this is to introduce profits or weights in a satisfaction problem and define an objective function such that the problem becomes an optimisation problem. The original satisfaction problem forms P_1 and the optimisation forms P_2. But although these problems are somewhat artificial there are real-world problems that naturally have weak symmetries. One example is from automated manufacturing that will be explained later.
Depending on the decomposition even standard problems have weak symmetries.

At the moment it is not investigated any further whether weak symmetries can be detected automatically with methods of automatic symmetry detection like graph isomorphism. Basically the problem is that the problem is not symmetric. A possible way would be to relax constraints and/or variables and check whether the resulting problem is symmetric. Most likely this does not have to be done for all subsets of constraints or variables. Considering the hyper-graph of the variables as nodes and the constraints as edges this may yield promising elements to relax. But this is just an assumption at this stage of research.

In the following we will state some problems containing weak symmetries.

3.1 The Magic Square Problem

With the proper decomposition even standard problems like the magic square problem contain weak symmetries.

In the magic square problem the task is to assign the numbers $1, \ldots, n^2$ to a $n \times n$ square M, whereby n is called the *order* of the magic square such that the sum of each row, column and the two diagonals equal the same number. This number is called the magic number m of the square and is computed by $m = \frac{n^3+n}{2}$ for each magic square of order n. The range of the columns and rows is denoted by $N = \{1, \ldots, n\}$.

The problem decomposes as following: In P_1 the task is to find an assignment of the square such that the sum of each row and column equals m. In P_{sym} SymVars are introduced for the columns (SymCol) and the rows (SymRow) and they are permuted to constitute a different square. In P_2 it is checked whether the (permuted) assignment also respects the diagonal constraints.

The idea is to find an assignment of the square relaxing the diagonal constraints first (P_1). Then consider a row and/or column permutation of the problem (P_{sym}) and check whether this permutation also satisfies the diagonal constraints to state a valid magic square (P_2).

In this decomposition the row and column permutations are weak symmetries which can be broken on P_1 since we use SymVars.

Although this decomposition seems very unintuitive first results look promising. In this decomposition it is easier to find an initial assignment for the square which could help in finding a solution early.

3.2 Weighted Magic Square Problem

The problem is basically the same as stated above.

In the weighted magic square problem a weight for each field of the square is introduced. When a number is assigned to a field M_{ij} its value is multiplied by this weight. The total value of the square is the sum of all values of the square. The task is to find a magic square with the highest total value.

The problem decomposes as following: In P_1 the task is to find an assignment of the square such that the sum of each row and column equals m. In P_{sym} SymVars are introduced for the columns (SymCol) and the rows (SymRow) and they are permuted to constitute a different square. In P_2 it is checked whether the assignment does also respect the diagonal constraints (states a magic square) and the overall value of the square is determined.

In this problem all symmetries (row and column permutations and reflections and rotations of the square) are weak symmetries which can be broken on P_1.

3.3 Example from Automated Manufacturing

In this problem a mounting complex consists of several mounting machines and have to mount electrical components – so called mounting tasks – on PC Boards. The task is to optimize the throughput rate of the mounting complex which means to find an evenly distribution of the mounting tasks to the mounting machines.

To mount a mounting task on a machine it must be placeable. A mounting task is placeable if two things holds:

1. The according mounting task must be visible on the machine
2. The corresponding component type of the mounting task must be assigned to the mounting machine

The first feature changes from machine to machine such that not every mounting machine can mount the same mounting task. This information is given as input. The second features is part of the problem. A setup consisting of several component types has to be assigned to each mounting machine.

A feasible setup to one machine is also feasible for each other machine. Therefore permuting the setups on the machines is a symmetry. But since a permutation of the setups results in a different place-ability of the machines the permutation is a weak symmetry.

The problem decomposes as follows: P_1 consists of determining a setup for each machine. In P_{sym} a permutation of the setups on the mounting complex is determined. P_2 consists of a distribution of the mounting tasks to the machines.

For each setup a SymVar is introduced. After all setups are determined in P_1 by assigning the SymVars it is clear on which mounting machine a setup is assigned. Depending on this assignment the mounting task distribution can be determined.

4 Outlook

We introduced the definitions for weakly decomposable problems and weak symmetries and presented some problems that have weak symmetries. Standard problems with the proper decomposition may have weak symmetries as well as extensions of standard problems and many real-world problems.

First results indicate that the use of SymVars is fruitful in most applications. Nonetheless more research has to be done in this fields to indicate and characterise problems and applications where the SymVar approach can be used most efficiently.

References

1. Peter Gregory *Almost–Symmetry in Planning* SymNet Workshop on Almost-Symmetry in Search, New Lanark, 2005
2. Alastair Donaldson *Partial Symmetry in Model Checking* SymNet Workshop on Almost-Symmetry in Search, New Lanark, 2005
3. Roland Martin *Approaches to Symmetry Breaking for Weak Symmetries* SymNet Workshop on Almost-Symmetry in Search, New Lanark, 2005
4. Warwick Harvey *Symmetric Relaxation Techniques for Constraint Programming* SymNet Workshop on Almost-Symmetry in Search, New Lanark, 2005
5. Z. Kiziltan, B. Hnich *Prob031: Rack Configuration Problem* http://4c.ucc.ie/ tw/csplib/prob031
6. Hosted by Guenter Stertenbrink *Computing Magic Knight Tours* http://magictour.free.fr
7. Compiled by George Jelliss *Knight's Tour Notes* http://www.ktn.freeuk.com