An approach to Symmetry Breaking in Distributed Constraint Satisfaction Problems

Roland Martin[1] and David A. Burke[2]

[1] SAF AG, Simulation, Analysis and Forecasting
and Darmstadt University of Technology Algorithmics Group, Germany
[2] Centre for Telecommunications Value-chain Research
and Cork Constraint Computation Centre,
Dept. of Computer Science, University College Cork, Ireland
martin@algo.informatik.tu-darmstadt.de, d.burke@4c.ucc.ie

Abstract. Symmetry breaking in Distributed Constraint Satisfaction Problems (DisCSPs) has, to the best of our knowledge, not been investigated. In this paper, we show that most symmetries in DisCSPs are actually weak symmetries, and we demonstrate how weak symmetry breaking techniques can be applied to DisCSPs.

1 Introduction

In Multi-agent Systems, many applications involve solving problems that are naturally distributed over a set of agents, e.g, in a disaster rescue scenario, multiple agents with interdependent plans need to have their task schedules coordinated [1]. Further examples include coordinating vehicle schedules in a transport logistics problem [2], or staff timetabling in large organisations with multiple departments [3]. One technique for modelling and solving these problems is the Distributed Constraint Satisfaction Problem (DisCSP) [4] formulism, which is an extension of the Constraint Satisfaction Problem (CSP). DisCSP algorithms solve the problem in its natural distributed state searching for globally acceptable solutions while minimising communcation between agents.

Symmetry breaking [5] has been shown to provide great benefits when solving centralised CSPs. By identifying assignments to a problem that are symmetrically equivalent, large areas of the search space can be eliminated by considering only one assignment taken from each equivalence set. Weak symmetries are a special case of symmetries that act only on a subset of the variables and the weakly symmetric solutions satisfy only a subset of the constraints of the problem. Weak symmetries occur in many real-world problems such as planning, scheduling and model checking [6, 7]. If a CSP is distributed among several agents then each agent only has access to a subset of the variables and the constraints. If the subproblem of an agent contains a symmetry, two solutions that are symmetric for the agent may not be symmetric in the context of the larger global problem, i.e. the solutions may not be equivalent for other agents in the problem, thus the symmetry is weak. That means the symmetries of the agent cannot be broken since the agent does not know whether these solutions are symmetric for other agents. A modelling approach introduced in [8] allows weak symmetries to be broken without losing solutions. In this paper we will show how this approach can be applied to DisCSPs which represents the first method of symmetry breaking in DisCSPs.

2 Distributed Constraint Satisfaction Problem

A Distributed Constraint Satisfaction Problem consists of a set $A = \{a_1, a_2, ..., a_n\}$ of $agents$, and for each agent a_i, a set $X_i = \{x_{i1}, x_{i2}, \ldots, x_{im_i}\}$ of *variables* it controls, such that $\forall i \neq j \ X_i \cap X_j = \emptyset$.Each variable x_{ij} has a corresponding domain D_{ij}. $X = \bigcup X_i$ is the set of all variables in the problem. $C = \{c_1, c_2, \ldots, c_t\}$ is a set of *constraints*. Each c_k has a *scope* $s(c_k) \subseteq X$. The *agent scope*, $a(c_k)$, of c_k is the set of agents that c_k acts upon: $a(c_k) = \{a_i : X_i \cap s(c_k) \neq \emptyset\}$. An agent a_i is a $neighbour$ of an agent a_j if $\exists c_k : a_i, a_j \in a(c_k)$. For each agent a_i, $p_i = \{x_{ij} : \forall c \ x_{ij} \in s(c) \rightarrow s(c) \subseteq X_i\}$ is its *private* variables – variables which are not constrained by other agents' variables – and $e_i = X_i \setminus p_i$ is its *public* variables – variables that have constraints with other agents. A *global assignment*, is the selection of one value for each variable in X. A *local assignment*, to an agent a_i, is an assignment to X_i. A solution to a DisCSP is a global assignment such that no constraints are violated. The solution process, however, is restricted: each agent is responsible for the assignment of its own variables, and thus agents must communicate with each other in order to find a global solution.

Asynchronous Backtracking (ABT) [4] is an asynchronous and distributed backtracking algorithm for solving DisCSPs. Agents act concurrently without any global control. A priority order is first created over the agents. Then, higher priority agents send *ok?* messages, containing a proposed assignment of their public variables, to lower priority agents. When an agent receives an *ok?* message it adds this assignment to its current *agent_view*, which is its belief of what values are assigned to the other agents' variables. It then searches for a solution to its subproblem that is consistent with this *agent_view*. If it finds a consistent solution, it sends its new assignment of its public variables in an *ok?* message to lower priority agents. If it can find no consistent assignment, it sends a *nogood* message specifying the assignment(s) that are causing the conflict to the lowest priority agent involved in the conflict (other than itself). The search progresses until a consistent global solution is found or the problem is proved infeasible.

To search for a consistent local assignment to an agent's subproblem, a centralised CSP solver can be used. To use symmetry breaking as part of this local solving process is difficult because: (i) solutions that are symmetric for one agent's subproblem but that contain different assignments to the agent's public variables may not be equivalent with regards to the global problem; and (ii) no single agent has a complete view of all variables and all constraints in the global problem. Since only an agent's public variables affect the other agents in the problem, symmetry breaking can be used only if all solutions in an equivalence set have identical assignments to the agent's public variables. However, if this is not the case it is not possible to break the symmetry without risking losing solutions. In Section 4, we propose the first steps in tackling this problem.

3 Weak Symmetry

If a CSP P has a subproblem P' where P' has a symmetry π that P does not then π is called a *weak symmetry* on P (see [8] for a more formal definition). The challenge in weak symmetry breaking is to not lose solutions when breaking the symmetries that exist in a subproblem. P' may contain several solutions to it that are symmetrical, however, although a solution s to the subproblem P' may be symmetrical to another solution

s' of the subproblem P', the solutions may not be equivalent with respect to the entire problem P. Therefore we cannot discard solutions as readily as in regular symmetry breaking. We need a way to represent all these solutions in the search process if we want to break the symmetry.

To do this, we use a modelling approach proposed in [8] that works by introducing additional variables called *SymVars* into a problem that can represent all solutions of an equivalence class. On these variables we state constraints that model the weak symmetry. The weak symmetry on P' can then be broken by any desired symmetry breaking method. When solving the problem, we first search for a solution for the subproblem P'. When a solution s to P' is found we instantiate the *SymVars* and then attempt to extend this solution to P. If a conflict is detected, we backtrack on the *SymVars* to find an equivalent solution to s (as opposed to searching for a new solution to P'). We then attempt to extend this new solution to P. Only if no permutation of s produces a feasible solution do we backtrack and search for a new solution that satisfies P'.

4 Breaking Weak Symmetries in DisCSPs

To demonstrate how we can break weak symmetries in DisCSP, we artificially construct a distributed magic square problem. In the magic square problem, the numbers $1, \ldots, n^2$ have to be assigned to a $n \times n$ square such that the sum of the numbers in each row, in each column and in both main diagonals are equal. The value m for this sum necessarily satisfies $m = \frac{n^3+n}{2}$. In our distributed scenario (Figure 1)[3], agent A owns all variables that make up a magic square but for its particular local subproblem it is only interested in enforcing row and column constraints. Agent B, holds variables that are constrained to be equal to the diagonal and anti-diagonal variables of agent A. Furthermore agent B will enforce the diagonal and anti-diagonal constraints on its variables. This can be seen as an instance of a multi-agent agreement problem where agents need to agree on the values of public variables that are bound by equality constraints, considering their own private internal variables and constraints[4].

We can apply weak symmetry breaking techniques on the resulting problem. Consider the column symmetries of agent A's subproblem. Any solution found that satisfies the row and column constraints can be permuted by changing the order of the columns to produce $n!$ symmetrical solutions. This symmetry can be broken by adding a constraint that orders the columns such that the first element in a column is less than the first element of the following column - thus, only one solution of each equivalence set will be found. However, in the DisCSP case, we can not afford to lose these equivalent

[3] We use a fictional distributed scenario of this well known problem in order to easily and clearly describe our approach. The techniques we describe are equally applicable to real world planning and scheduling scenarios with multiple agents, each with large subproblems.

[4] In a DisCSP each agent owns and chooses assignments for its own independent set of variables. In this example the equality constraints ensure that it is beneficial for the agents to set their public variables identically, but there are still two sets of independent variables, i.e. the variables are not shared. In the general case, public variables may be bound by any type of constraint, not just equality constraints.

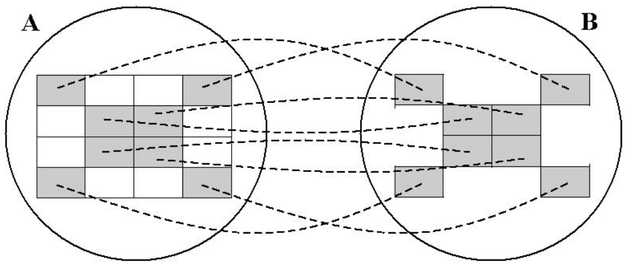

Fig. 1. DisCSP formulation of Magic Square problem. Agent A contains all the variables of the square and will enforce row and column constraints on these. Agent B contains variables that are constrained to be equal to the diagonal variables of Agent A (dashed lines indicate equality constraints) and will enforce the diagonal and anti-diagonal constraints on its variables.

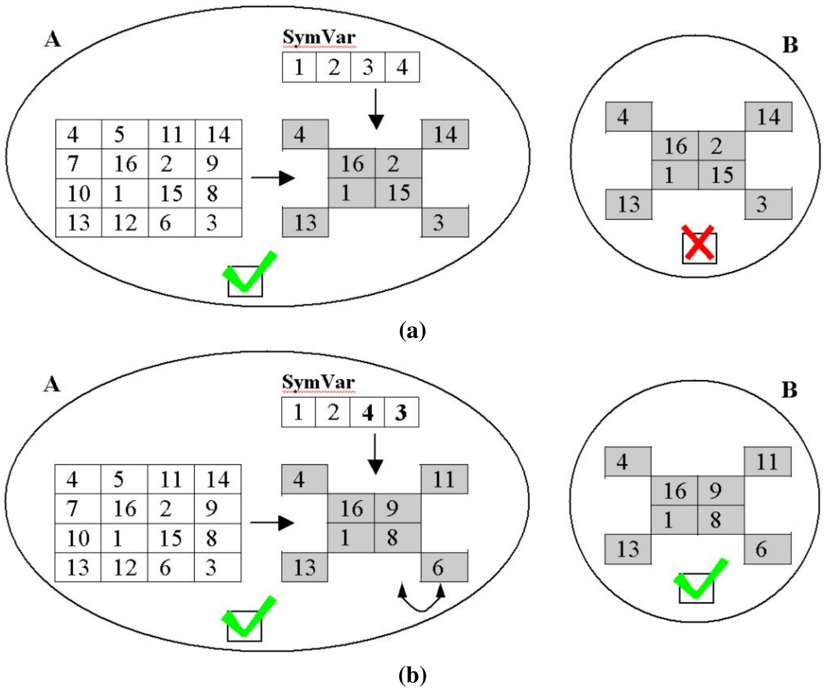

Fig. 2. Magic square: $n = 4, m = 34$. (a) - by introducing *SymVars* agent A can break the column symmetry of its original subproblem. (b) - equivalent solutions can then be generated using the *SymVars* to find a permutation that is also valid for agent B's constraints.

solutions. It is possible that one valid solution found for agent A can not be extended to a valid solution in agent B, while an equivalent solution can be.

The challenge we face is how to enforce symmetry without losing solutions that may be required by other agents in the problem. To do this, we introduce a *SymVar* for each column to agent A to represent the permutations of a solution to its subproblem (Figure 2). In our example, an assignment to the *SymVars* will represent one particular permutation of the columns of a solution, and the *SymVars* will be constrained to be *alldifferent*. We also introduce variables to represent the permuted solution *projected* on the agent's public variables. We consider an example execution of ABT and assume that agent A is assigned a higher priority than agent B. Agent A will search for a solution to the variables of its magic square (using column symmetry breaking) observing the row and column constraints (Figure 2 (a)). Next the *SymVars* are assigned, which in turn leads to an assignment of the new projected public variables. Agent A will send the assignment of its public variables to agent B. Agent B's public constraints require that its assignments to its variables must be equal to the corresponding public variables of agent A. However, by enforcing this it will not have a solution consistent with its internal constraints, i.e. the diagonal and anti-diagonal constraints are not satisfied. Therefore, it sends a *nogood* to agent A and agent A will add a constraint to its subproblem that rules out this assignment. Agent A will then select a new solution using its *SymVars*, i.e. instead of searching for a new solution to its magic square, it just finds a new assignment to its *SymVars*, which leads to a new assignment to the projected public variables (Figure 2 (b)). A new non-symmetric solution for the agent's internal problem will *only* be searched for once all symmetrical solutions have been exhausted. In our example, the new permutation of the original solution produces an assignment to the public variables that allow agent B's constraints to also be satisfied, and so a global solution has been found.

The benefit of this approach is that finding a new assignment to the *SymVars* is much less expensive than searching for a new solution to the magic square. We can break symmetries in a standard way, while the *SymVars* will ensure that equivalent solutions can be generated if required.

5 Conclusions and Outlook

We have showed that most symmetries in DisCSPs are actually weak symmetries and have applied fruitfully a weak symmetry breaking approach to DisCSPs. This is the first symmetry breaking approach to DisCSPs as far as we know. We demonstrated the approach using the magic square problem modelled as a DisCSP. This paper represents our initial investigations into this issue and future work will focus on implementing and evaluating the approach using state-to-the-art DisCSP algorithms.

6 Acknowledgements

D. Burke is supported by Science Foundation Ireland under Grant No. 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain Research (CTVR). The authors would like to thank the reviewers for useful comments.

References

1. Sultanik, E.A., Modi, P.J., Regli, W.C.: Constraint propagation for domain bounding in c-taems task scheduling. In: Proc. CP. (2006) 756–760
2. Calisti, M., Neagu, N.: Constraint satisfaction techniques and software agents. In: Proc. Agents and Constraints Workshop, AIIA. (2004)
3. Meisels, A., Kaplansky, E.: Scheduling agents - distributed timetabling problems(disttp). In: Proc. PATAT. (2002) 166–177
4. Yokoo, M., Hirayama, K.: Algorithms for distributed constraint satisfaction: A review. Autonomous Agents and Multi-Agent Systems **3**(2) (2000) 185–207
5. Gent, I.P., Petrie, K.E., Puget, J.F.: Symmetry in constraint programming. In: Handbook of Constraint Programming. F. Rossi and P. van Beek and T. Walsh (2006)
6. Gregory, P.: Almost–symmetry in planning. In: Proc. SymNet Workshop on Almost-Symmetry in Search. (2005) 14–16
7. Donaldson, A.: Partial symmetry in model checking. In: Proc. SymNet Workshop on Almost-Symmetry in Search. (2005) 17–21
8. Martin, R.: The challenge of exploiting weak symmetries. In: Proc. CSCLP. (2005) 149–163