Exploiting Constraints

Toby Walsh

NICTA and UNSW, Sydney, Australia

Abstract. Constraints can be exploited in paradigms outside of constraint programming. In particular, powerful global constraints can often be decomposed into small primitives and these decompositions can simulate complex propagation algorithms that perform sophisticated inference about a problem. We illustrate this approach with examples of exploiting constraints in propositional satisfiability (SAT), pseudo-Boolean (PB) solving, integer linear programming (ILP) and answer set programming (ASP).

1 Introduction

Constraint programming is a successful technology used to solve a wide range of combinatorial optimisation problems involing scarce or expensive resources like routing and scheduling [1]. Central to this success are global constraints. These capture common modelling patterns [2] (e.g. "all these activities occur on the same machine so must take place at different times") and take advantage of powerful and efficient propagation algorithms to reason about possible solutions (e.g. "we have five activities taking placing on four machines so, by a pigeonhole argument, we must take at least two time periods"). However, there are other paradigms that have other strengths. For example, satisfiability solvers typically provide sophisticated methods for recovering from branching mistakes like nogood learning and restarts. As a second example, answer set programming provides an even richer modelling language for modelling and reasoning about concepts like the transitive closure of a relation. It is therefore useful to develop methods for exploiting global constraints in these other paradigms. In this paper, we survey recent work in this area which shows that global constraints can often be exploiting in these other paradigms by means of carefully designed decompositions. On the other hand, we also describe recent work in this area which clearly identifies the limits of this approach, demonstrating that there are certain global constraints that cannot be replaced by any polynomial sized decomposition.

2 An example

One of the oldest and most useful global constraint is the ALLDIFFERENT constraint [3, 4]. This ensures that a set of variables are pairwise different. A simple decomposition of the ALLDIFFERENT constraint is into a clique of binary inequalities. However, this decomposition usually hinders inference. Consider the following running example: we have three variables, $X_1 \in \{0, 1, 2\}$ and X_2, $X_3 \in \{1, 2\}$ that must take different values to each other. Then any binary inequality, say $X_1 \neq X_3$ is domain consistent

(that is, for every value in the domain of X_1 there is a value available in the domain of X_3 that satisfies the constraint and vice versa). A more complex decomposition [5] will provide a more "global" view that simulates the efficient propagation algorithms that have been developed for the global ALLDIFFERENT constraint [6–9]. We introduce 0/1 variables, A_{ilu} which by means of the following channeling constraints represent whether X_i takes a value in the interval $[l, u]$:

$$A_{ilu} = 1 \Leftrightarrow X_i \in [l, u]$$

To ensure that $X_i \neq X_j$ for any $i < j$, we add constraints that enforce the Hall interval property that the total number of variables taking values within any interval is no more than the size of that interval:

$$\sum_{i=1}^{n} A_{ilu} \leq u - l + 1$$

Consider our running example again and the interval $[1, 2]$. Now A_{212} and A_{312} are both 1 as X_2 and X_3 must take their value from within this interval. Since $\sum_{i=1}^{3} A_{i12} \leq 2$, it follows that $A_{112} = 0$. That is, X_1 cannot take a value from the interval $[1, 2]$ and must instead be set to 0. In fact, bound consistency on this decomposition will ensure bound consistency on the global ALLDIFFERENT constraint [5], simulating the actions of a complex propagation algorithm. This decomposition can be readily used in a pseudo-Boolean solver.

3 Encoding domains

In many decompositions, we reduce the problem to one on 0/1 variables. This allows us both to reason about individual assignments (e.g. "is X_3 set to 2 or not?") and to use the decomposition in a different type of solver (e.g. in a satisfiability solver). There are several encodings used to map multi-valued domains onto 0/1 variables depending on the inference we want to simulate in the decomposition.

Direct encoding: We set $B_{ij} = 1$ iff $X_i = j$ [10]. This gives access to the individual values to be assigned to each variable but often hinders propagation.

Order encoding: We set $B_{ij} = 1$ iff $X_i \leq j$. This looks superficially similar to the direct encoding (e.g. it requires the same number of 0/1 variables) but can offer important inferential advantages. For instance, this encoding is used in the decomposition of the domain consistency propagator for the lexicographical ordering constraint [11].

Interval encoding: We set $A_{ilu} = 1$ iff $X_i \in [l, u]$. Such an encoding is used in decompositions of many counting and occurrence constraints (e.g. ALLDIFFERENT, GCC, NVALUE [5, 12]).

4 Encoding into ASP

One paradigm in which we have had considerable success in exploiting constraints is answer set programming (ASP). This is a powerful modelling and solving paradigm

that offers a number of advantages. For instance, ASP supports recursive defintions which permits us to model easily concepts like the transitive closure of a relation. As a second example, ASP supports default negation which permits us to model easily a range of real world problems. By using suitable decompositions, we can add global constraints to this list of advantages of ASP [13–15]. For instance, we have added global constraints to ASP which allow for the specification of constraints in terms of automata which accept just valid assignments for sequences of variables [16]. Such constraints are useful in a wide range of scheduling, rostering and sequencing problems to ensure certain patterns do or do not occur over time. For example, we may wish to ensure that anyone working three night shifts then has two or more days off. Such a constraint can easily be expressed using a regular language.

5 Advantages of decomposition

There are several advantages to modelling global constraints by sophisticated decompositions like these.

Exporting constraints: These decompositions can often be easily used in a different solving paradigm. For example, we have used such decomposition in both state of the art pseudo-Boolean and answer set solvers. However, we could also have used them in an ILP solver.

Combining constraints: These decompositions often introduce new variables that describe internal state. When multiple constraints are posted, we can often share such variables to increase propagation [17].

Extending constraints: These decompositions may enable or suggest interesting extensions of our solver. For instance, our decompositions of the ALLDIFFERENT constraint suggest learning nogoods based on small Hall intervals.

Branching: These decompositions open out the constraint propagation algorithm. Previously such algorithms were black boxes to the rest of the solver. However, the internal state of the propgagator may be useful to guide branching and other decisions.

Incremental propagation: Decompositions are naturally incremental. When a problem changes (e.g. we assign a variable), only those parts of the decomposition touched need wake up.

On the other hand, there are two features of decompositions that may limit their usefulness:

Space complexity: By their nature, the space complexity of a decomposition tends to equal the time complexity. This is because we use techniques like unit propagation on the decomposition which are linear in the size of the problem. This may be problematic when, for example, domains are large and the propagator for the global constraint takes quadratic or cubic time in the domain size.

Time complexity: By their nature, the time complexity of reasoning with the decomposition tends to be the same in the best case as in the worst case. This is because the decomposition must anticipate all possible inferences. This means that decompositions tend to perform well when they are simulating propagators that use dynamic programming which naturally have this property.

6 Limits of decomposition

These concerns about the time and space complexity of decompositions can be made rather concrete. There are certain global inferences that *cannot* be simulated *effectively* using decompositions. In particular, we have proved that there is no polynomial sized decomposition of the global ALLDIFFERENT constraints into conjunctive normal form (CNF) with the property that unit propagation on the decomposition enforces domain consistency on the original global constraint [18]. The proof uses lower bounds on the size of monotone Boolean circuits. We show that there is a polynomial sized decomposition of a constraint propagator into CNF if and only if the propagator can be computed by a polynomial size monotone Boolean circuit. The super-polynomial lower bound on the size of a Boolean circuit for computing a perfect matching in a bipartite graph thus gives a super-polynomial lower bound on the size of a CNF decomposition of the domain consistency propagator for the global ALLDIFFERENT constraint. It follows that domain consistency propagators for other global constraints which generalize ALLDIFFERENT like GCC and NVALUE also can be effectively simulated using decompositions.

7 Other decompositions

There are several other global constraints that have been shown to be effectively decomposable.

SEQUENCE**:** The global SEQUENCE constraint is used in modelling car sequencing, rostering, scheduling and related problems. Several decompositions for this constraint have been introduced, two of which successively improved the best known asymptotic bound for propagating the constraint [19, 20]. Some of these decompositions are based on mapping the problem into a linear program that represent a network flow.

PRECEDENCE**:** The global PRECEDENCE constraint [21] is used to break value symmetry [22–24]. A simple, linear decomposition into ternary constraints simulates the domain consistency propagator [25].

ROOTS**,** RANGE**:** Many global constraints dealing with counting and occurrences can be decomposed into two primitives, ROOTS and RANGE [26]. A simple flow-based decomposition can be used to propagate the RANGE constraint [27]. Whilst propagating the ROOTS constraint completely is NP-hard in general [28], many of the cases needed in practice are polynomial based on a simple decomposition [29].

SLIDE**:** The global SLIDE constraint is a "meta-constraint" that slides a given constraint down a sequence of variables [30, 31]. It can be used to model many other global constraints and is useful to specify rostering and other problems (e.g. in any 7 day time window, I must have 2 days off). As with ROOTS, propagating the constraint completely is NP-hard in general [32]. However, there are many cases met in practice (e.g. when the constraint being slid is of fixed arity), when decomposition is able to propagate the global constraint completely.

TABLE**:** The TABLE constraint can model any arbitrary relation. It is useful in configuration and related problems for expressing product compatability. Bacchus has

given a decomposition into CNF on which unit propagation achieves domain consistency [33].

8 Conclusion

We have surveyed recent work in decomposing global constraints. Such decompositions permit us to exploit global constraints in paradigms outside of constraint programming like ASP and SAT. On the other hand, there also exists limits to what can be achieved with decompositions. For instance, we cannot effectively simulate a domain consistency propagators for a global constraint like ALLDIFFERENT. There are many interesting open questions in this area. For example, we have not been able to design a decomposition that effectively simulates the doman consistency propagator for the LEXCHAIN constraint [34]. However, we have not been able to prove that no such decomposition exists. We conjecture that the latter holds.

References

1. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Foundations of Artificial Intelligence. Elsevier (2006)
2. Walsh, T.: Constraint patterns. In 9th Int. Conf. on Principles and Practices of Constraint Programming (CP-2003), (2003)
3. Lauriere, J.: Alice: A language and a program for solving combinatorial problems. Artificial Intelligence **10** (1978) 29–127
4. Régin, J.C.: A filtering algorithm for constraints of difference in CSPs. In: Proc. of the 12th National Conf. on AI, AAAI (1994) 362–367
5. Bessière, C., Katsirelos, G., Narodytska, N., Quimper, C.G., Walsh, T.: Decompositions of all different, global cardinality and related constraints. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, (2009)
6. Leconte, M.: A bounds-based reduction scheme for constraints of difference. In: Proc. of Second Int. Workshop on Constraint-based Reasoning (Constraint-96). (1996)
7. Puget, J.: A fast algorithm for the bound consistency of alldiff constraints. In: 15th National Conf. on Artificial Intelligence, AAAI (1998) 359–366
8. Mehlhorn, K., Thiel, S.: Faster algorithms for bound-consistency of the sortedness and the alldifferent constraint. In: Proc. of Sixth Int. Conf. on Principles and Practice of Constraint Programming (CP2000), (2000) 306–319
9. Lopez-Ortiz, A., Quimper, C., Tromp, J., van Beek, P.: A fast and simple algorithm for bounds consistency of the alldifferent constraint. In: Proc. of the 18th Int. Joint Conf. on AI, IJCAI (2003)
10. Walsh, T.: SAT v CSP. In: 6th Int. Conf. on Principles and Practices of Constraint Programming (CP-2000), (2000) 441–456
11. Gent, I., Prosser, P., Smith, B.: A 0/1 encoding of the GACLex constraint for pairs of vectors. In: Proc. of ECAI-2002 Workshop on Modelling and Solving Problems with Constraints. (2002)
12. Bessiere, C., Katsirelos, G., Narodytska, N., Walsh, C.G.Q.T.: Decomposition of the NValue constraint. In: Proc. of the 16th Int. Conf. on the Principles and Practice of Constraint Programming (CP 2010). (2010)
13. Drescher, C., Walsh, T.: A translational approach to constraint answer set solving. Theory and Practice of Logic Programming **10**(4-6) (2010) 465–480

14. Drescher, C., Walsh, T.: Modelling GRAMMAR constraints with answer set programming. In: Proc. of the 27th Int. Conf. on Logic Programming (ICLP 2011). (2011)
15. Drescher, C., Walsh, T.: Translation-based constraint answer set solving. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence, IJCAI (2011)
16. Pesant, G.: A regular language membership constraint for finite sequences of variables. In: Proc. of 10th Int. Conf. on Principles and Practice of Constraint Programming (CP2004), (2004) 482–295
17. Bessiere, C., Katsirelos, G., Narodytska, N., Quimper, C.G., Walsh, T.: Propagating conjunctions of alldifferent constraints. In: Proc. of the Twenty-Fourth AAAI Conf. on Artificial Intelligence (AAAI 2010), AAAI (2010)
18. Bessière, C., Katsirelos, G., Narodytska, N., Walsh, T.: Circuit complexity and decompositions of global constraints. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI (2009)
19. Brand, S., Narodytska, N., Quimper, C.G., Stuckey, P., Walsh, T.: Encodings of the sequence constraint. In: 13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007), (2007)
20. Maher, M., Narodytska, N., Quimper, C.G., Walsh, T.: Flow-based propagators for the SEQUENCE and related global constraints. In: 14th Int. Conf. on Principles and Practices of Constraint Programming (CP-2008), (2008) 159–174
21. Law, Y., Lee, J.: Global constraints for integer and set value precedence. In: Proc. of 10th Int. Conf. on Principles and Practice of Constraint Programming (CP2004), (2004) 362–376
22. Walsh, T.: General symmetry breaking constraints. In: 12th Int. Conf. on Principles and Practices of Constraint Programming (CP-2006), (2006)
23. Walsh, T.: Breaking value symmetry. In: 13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007), (2007)
24. Walsh, T.: Breaking value symmetry. In: Proc. of the 23rd National Conf. on AI, AAAI (2008) 1585–1588
25. Walsh, T.: Symmetry breaking using value precedence. In: Proc. of the 17th European Conf. on Artificial Intelligence (ECAI-2006), IOS Press (2006)
26. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: The range and roots constraints: Specifying counting and occurrence problems. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI (2005) 60–65
27. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: The range constraint: Algorithms and implementation. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Third Int. Conf. (CPAIOR 2006). (2006) 59–73
28. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. In: Proc. of the 19th National Conf. on AI, AAAI (2004)
29. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: The ROOTS constraint. In: 12th Int. Conf. on Principles and Practices of Constraint Programming (CP-2006), (2006)
30. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C.G., Walsh, T.: Reformulating global constraints: the slide and regular constraints. In: Abstraction, Reformulation, and Approximation: Proc. of 4th Int. Symposium. (2007)
31. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: SLIDE: a useful special case of the CardPath constraint. In: Proc. of the 18th European Conf. on Artificial Intelligence (ECAI-2008), IOS Press (2008)
32. Bessière, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. Constraints **12**(2) (2007) 239–259
33. Bacchus, F.: GAC via unit propagation. In: Proc. of 13th Int. Conf. on Principles and Practice of Constraint Programming (CP2007), (2007) 133–147
34. Carlsson, M., Beldiceanu, N.: Arc-consistency for a chain of lexicographic ordering constraints. Tech. rep. T2002-18, Swedish Institute of Computer Science (2002)