Beyond Finite Domains: the All Different and Global Cardinality Constraints

Claude-Guy Quimper[1] and Toby Walsh[2]

[1] School of Computer Science, University of Waterloo, Canada,
`cquimper@math.uwaterloo.ca`
[2] NICTA and UNSW, Sydney, Australia, `tw@cse.unsw.edu.au`

Abstract. We describe how the propagator for the All-Different constraint can be generalized to prune variables whose domains are not just simple finite integer domains. We show, for example, how it can be used to propagate set, multiset and tuple variables.

1 Introduction

Constraint programming has restricted itself largely to finding values for variables taken from finite integer domains. However, we might want to consider variables representing sets [11–13], multisets [14], ordered tuples, or other structures. These variable types reduce the space needed to represent possible domain values, improve the efficiency of constraint propagators and inherit all the usual benefits of data abstraction like ease of debugging and code maintenance.

As an example, consider the round robin sports scheduling problem (prob026 in CSPLib). In this problem, we wish to find a schedule satisfying a number of constraints including that a team never plays twice with another team. We therefore would like a propagator which works on an All-Different constraint posted on variables whose values are pairs of teams. In this paper, we consider how to efficiently and effectively implement the All-Different constraint on variables whose values are sets, multisets or tuples. Due to space restrictions, we omit proofs. A longer version of the paper is available as a technical report.

2 Propagators for the All-Different Constraint

Propagating the All-Different constraint involves removing from the domain of variables those values that cannot be part of a consistent assignment. To design his propagator, Leconte [16] introduced the concept of *Hall set* based on Hall's work [1].

Definition 1. *A* Hall set *is a set H of values such that the number of variables whose domain is contained in H is equal to the cardinality of H. More formally, H is a Hall set if and only if $|H| = |\{x_i \mid dom(x_i) \subseteq H\}|$.*

To enforce domain consistency, it is necessary and sufficient to detect every Hall set H and remove its values from the domains that are not fully contained in H. Régin's propagator [4] uses matching theory to detect Hall sets. Leconte [16], Puget [17], López-Ortiz et al. [9] use simpler ways to detect Hall intervals and achieve weaker consistencies.

3 Beyond integer variables

A propagator designed for integer variables can be applied to any type of variable whose domain can be enumerated. For instance, let the following variables be sets whose domains are expressed by a set of required values and a set of allowed values. $\{\} \subseteq S_1, S_2, S_3, S_4 \subseteq \{1, 2\}$ and $\{\} \subseteq S_5, S_6 \subseteq \{2, 3\}$. Variable domains can be expanded as follows: $S_1, S_2, S_3, S_4 \in \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$ and $S_5, S_6 \in \{\{\}, \{2\}, \{3\}, \{2, 3\}\}$. By enforcing GAC on the All-Different constraint, we obtain $S_1, S_2, S_3, S_4 \in \{\{\}, \{1\}, \{2\}, \{1, 2\}\}$ and $S_5, S_6 \in \{\{3\}, \{2, 3\}\}$. We can now convert the domains back to their initial representation. $\{\} \subseteq S_1, S_2, S_3, S_4 \subseteq \{1, 2\}$ and $\{3\} \subseteq S_5, S_6 \subseteq \{2, 3\}$.

This technique always works but is not tractable in general since variable domains might have exponential size. For instance, the domain of $\emptyset \subseteq S_i \subseteq [1, n]$ contains 2^n elements. The following important lemma allows us to ignore such variables and focus just on those with "small" domains.

Lemma 1. *Let n be the number of variables and let F be a set of variables whose domains are not contained in any Hall set. Let $x_i \notin F$ be a variable whose domain contains more than $n - |F|$ values. Then $dom(x_i)$ is not contained in any Hall set.*

Lemma 1 helps us to find variables F whose domain cannot be contained in a Hall set. Algorithm 1 prunes the domains of n variables and ensures that domains larger than n do not slow down the propagation.

```
  F ← ∅
1 for x_i ∈ X do  if |dom(x_i)| > |X| − |F| then  F ← F ∪ {x_i}
2 Expand domains of variables in X − F.
  Propagate the All-Different constraint on variables X − F and find Hall sets H.
  for x_i ∈ F do  dom(x_i) ← dom(x_i) − H
3 Collapse domains of variables in X − F.
```

Algorithm 1: All-Different propagator for variables with large domains

To apply our new techniques, three conditions must be satisfied by the representation of the variables: computing the size of the domain must be tractable (Line 1), domains must be efficiently enumerable (Line 2) and efficiently computed from an enumeration of values (Line 3). The next sections describe how different representations of domains for set, multiset and tuple variables meet these three conditions.

4 All-Different on sets

Several representations of domains have been suggested for set variables. The most common representations use a set of required elements lb and a set of allowed elements ub such that any set S satisfying $lb \subseteq S \subseteq ub$ belongs to the domain [11, 12]. The cardinality of $\mathrm{dom}(S)$ is $2^{|ub-lb|}$ and can be computed in constant time. Often, to represent more precisely the possible values, a cardinality variable C is added such that $|S| \in \mathrm{dom}(C)$. The size of the domain is then given by $\sum_{j \in \mathrm{dom}(C)} \binom{|ub-lb|}{j-|lb|}$ and this can be computed in $O(|\mathrm{dom}(C)|)$ steps.

To increase the expressiveness of the domain representation, Sadler and Gervet [6] suggest adding a lexicographic ordering constraint. We therefore say that $S_1 < S_2$ holds if S_1 comes before S_2 in a lexicographical order. The new domain representation now involves two lexicographic bounds $l \leq S \leq u$. To compute the size of such domains, we consider the binary vector representation where each bit of a vector corresponds to an element in $ub - lb$. The bit is set to 1 if the element belongs to the set and 0 otherwise. Let a and b be the binary vector representation of the lexicographical bounds l and u. Let $a - 1$ be the vector that lexicographically precedes a. Function f computes the number of binary vectors lexicographicaly smaller than or equal to s with k bits set to one. Assuming that $\binom{x}{y} = 0$ for any $y < 0$, the size of domain S is given by the following equations.

$$|\mathrm{dom}(S)| = \sum_{k \in C} \left(f(b, k) - f(a-1, k)\right) \tag{1}$$

$$f([s_m, \ldots, s_1], k) = \sum_{i=1}^{m} s_i \binom{i-1}{k - \sum_{j=i+1}^{m} s_j} + \delta(\boldsymbol{s}, k) \tag{2}$$

$$\delta([s_m, \ldots, s_1], k) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} s_i = k \text{ and } s_0 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Function f can be evaluated in $O(|ub-lb|)$ steps. The size of domain $\mathrm{dom}(S)$ therefore requires $O(|ub - lb||\mathrm{dom}(C)|)$ steps to compute.

We can enumerate the sets in $\mathrm{dom}(S)$ of cardinality k for each $k \in \mathrm{dom}(C)$. Based on the lexicographic bound l, we find the first set of cardinality k. Algorithm T from Knuth [8] provides subsequent sets. Proceeding this way results in a $O(|\mathrm{dom}(C)||ub - lb| + |\mathrm{dom}(S)|)$ algorithm. When there are no lexicographic bounds, the complexity can be reduced to $O(\max(|ub - lb|, |\mathrm{dom}(S)|))$.

5 All-Different on tuples

A tuple t is an ordered sequence of n elements that allows multiple occurrences. The most common way to represent the domain of a tuple is simply by associating an integer variable to each of the tuple components. A tuple of size n is therefore represented by n integer variables $x_1, \ldots, x_n$.

To apply an All-Different constraint to a set of tuples, a common solution is to create an integer variable t for each tuple. If each component x_i ranges from 0 to c_i exclusively, we add the channeling following constraint $t = \sum_i^n \prod_{j=i+1}^n c_j x_i$.

This technique suffers from either inefficient or ineffective channeling between variable t and the components x_i. Most constraint libraries enforce bound consistency on t. A modification to the domain of x_i does not necessarily affect t. Conversely, even if all tuples encoded in $\mathrm{dom}(t)$ have $x_i \neq v$, value v will most often not be removed from $\mathrm{dom}(x_i)$. On the other hand, enforcing domain consistency typically requires $O(|\mathrm{dom}(t)|)$ steps which can be time consuming when domains are large.

To address this issue, one can define a tuple variable whose domain is defined by the domain of its components. $\mathrm{dom}(t) = \mathrm{dom}(x_1) \times \ldots \times \mathrm{dom}(x_n)$. The size of such a domain is given by $|\mathrm{dom}(t)| = \prod_{i=1}^n |\mathrm{dom}(x_i)|$ which can be computed in $O(n)$ steps.

As Sadler and Gervet [6] did for sets, we can add lexicographical bounds to tuples $l \leq t \leq u$ in order to better express the values the domain contains.

Let $idx(v, x)$ be the number of values smaller than v in the domain of the integer variable x. Assuming $idx(v, x)$ has a running time complexity of $O(\log(|\mathrm{dom}(x)|))$, the size of the domain can be evaluated in $O(n+\log(|\mathrm{dom}(t)|))$ steps using $|\mathrm{dom}(t)| = 1+\sum_{i=1}^n \left((idx(u[i], x_i) - idx(l[i], x_i)) \prod_{j=i+1}^n |\mathrm{dom}(x_i)|\right)$

Algorithm M from Knuth [7] enumerates the domain of a tuple variable in lexicographical order. Assuming the domain of all component variables have the same size, this algorithm runs in $O(|dom(t)|)$ steps which is optimal.

6 All-Different on multi-sets

Unlike sets, multi-sets allow multiple occurrences of a same element. A multi-set can be represented by a tuple where each component indicates the multiplicity of an element in the multi-set. All algorithms explained in Section 5 can therefore be applied to multi-sets.

7 Indexing domain values

Propagators for the All-Different constraint, like the one proposed by Régin [4], need to store information about the values appearing in the domains of variables. When values are integers, a table T can store information related to value v in entry $T[v]$. Algorithm 1 ensures that no more than n^2 distinct values will be handled by the propagator. When these n^2 values come from a significantly larger set of values, table T becomes very sparse. To allow better direct access, we need to map the n^2 values to an index in the interval $[1, n^2]$. The trie data structure retrieves the value associated to a set, a multi-set, a tuple, or any other sequential data structure of length l in $O(l)$ steps. This technique permits existing propagators to work without a penalty for sparse domain values.

8 Conclusions

We have described how existing propagators for the ALL-DIFFERENT constraint can be generalized to prune variables whose domains are not just simple finite integer domains. In particular, we described how it can be used to propagate set, multi-set and tuple variables. This result can easily be generalized for the global cardinality constraint. Many other global constraints still remain to be generalized to deal with variables which are not just simple integer finite domains, as well as to variables of other types.

References

1. P. Hall. On representatives of subsets. *Journal of the London Mathematical Society*, pages 26–30, 1935.
2. J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2:225–231, 1973.
3. ILOG S. A. ILOG Solver 4.2 user's manual, 1998.
4. J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of the 12th National Conference on Artificial Intelligence*, pp 362–367, 1994.
5. K. Stergiou and T. Walsh. The difference all-difference makes. In *Proc. of the 16th Int. Joint Conference on Artificial Intelligence*, pages 414–419, 1999.
6. A. Sadler and C. Gervet Hybrid Set Domains to Strengthen Constraint Propagation and Reduce Symmetries In *In Proc. of the 10th Int. Conference on Principles and Practice of Constraint Programming*, pp 604–618, 2004.
7. D. Knuth, *Volume 4 of The Art of Computer Programming, Pre-Fascicle 2a: Generating all n-tuples*, http://www-cs-faculty.stanford.edu/~knuth/
8. D. Knuth, *Volume 4 of The Art of Computer Programming, Pre-Fascicle 3a: Generating all combinations*, http://www-cs-faculty.stanford.edu/~knuth/
9. A. López-Ortiz, C.-G. Quimper, J. Tromp, and P. van Beek. A fast and simple algorithm for bounds consistency of the alldifferent constraint. In *Proc. of the 18th Int. Joint Conference on Artificial Intelligence*, pages 245–250, 2003.
10. W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling *Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition* Cambridge Univ. Press, 1992
11. C. Gervet Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language. CONSTRAINTS journal 1(3) (1997) p. 191-244
12. J.-F. Puget Finite set intervals. In *Proc. of Workshop on Set Constraints*, CP1996.
13. T. Müller and M. Müller. Finite set constraints in Oz. In F. Bry,B. Freitag, and D. Seipel, editors, *13. Workshop Logische Programmierung*, pages 104–115, Technische Universität München, 17–19 September 1997.
14. T. Walsh Consistency and Propagation with Multiset Constraints: A Formal Viewpoint In *Proc. of the 9th International Conference on Principles and Practice of Constraint Programming*, 2003.
15. I.P. Gent and T. Walsh CSPLib: a benchmark library for constraints *Technical report APES-09-1999*, 1999.
16. M. Leconte. A bounds-based reduction scheme for constraints of difference. In *Proceedings of the Constraint-96 International Workshop on Constraint-Based Reasoning,* pp. 19–28, 1996
17. J.-F. Puget. A Fast Algorithm for the Bound Consistency of Alldiff Constraints. In *Proc. of the 15th National Conf. on Artificiel Intelligence*, pp 359–366, 1998