Constructive Cardinality

Nicolas Beldiceanu and Mats Carlsson

SICS, Lägerhyddsvägen 18, SE-75237 Uppsala, Sweden
{nicolas,matsc}@sics.se

August 14 2001
SICS Technical Report T2001:15
ISSN 1100-3154
ISRN: SICS-T--2001:15-SE

Abstract. We describe a set of necessary conditions that are useful for generating propagation algorithms for the *cardinality* operator as well as for over-constrained problems with preferences. Constructive disjunction as well as the entailments rules originally proposed for the *cardinality* operator can be seen as simple cases of these necessary conditions. In addition these necessary conditions have the advantage of providing more pruning.

Keywords: Combinatorial problems, *cardinality* constraint, constructive disjunction, soft constraints.

CONSTRUCTIVE CARDINALITY

Nicolas BELDICEANU and Mats CARLSSON

SICS, Lägerhyddsvägen 18, SE-75237 Uppsala, Sweden
{nicolas,matsc}@sics.se

We describe a set of necessary conditions that are useful for generating propagation algorithms for the *cardinality* operator as well as for over-constrained problems with preferences. Constructive disjunction as well as the entailments rules originally proposed for the *cardinality* operator can be seen as simple cases of these necessary conditions. In addition these necessary conditions have the advantage of providing more pruning.

Keywords: Combinatorial problems, *cardinality* constraint, constructive disjunction, soft constraints

1 Introduction

For definitions from constraint programming not given here we refer to [3]. A domain variable V is a variable that ranges over a finite set of integers; $\mathrm{dom}(V)$, $\min(V)$ and $\max(V)$ respectively denote the set of possible values, the minimum and the maximum values of V. A constraint $CTR(V_1,..,V_n)$ over an ordered set of variables $V_1,..,V_n$ is a relation over the cartesian product of the domains of $V_1,..,V_n$, while the negation $\neg CTR(V_1,..,V_n)$ is the complement with respect to that cartesian product. To any constraint or conjunction of constraints CT, we associate a 0-1 domain variable B_{CT} that is equal to 0 if CT does not hold, and 1 if CT holds. We note $\#CT = \max(B_{CT})$.

Since its introduction in 1991 within constraint programming, the *cardinality* operator [2] has been recognized as a generic combinator which was progressively integrated into several constraint systems. Its most general form is $\mathrm{cardinality}(C, \{CTR_1(V_{11},..,V_{1k_1}),..,CTR_n(V_{n1},..,V_{nk_n})\})$ where C is a domain variable and $\{CTR_1(V_{11},..,V_{1k_1}),..,CTR_n(V_{n1},..,V_{nk_n})\}$ is a collection of constraints over finite domain variables. The *cardinality* operator holds iff $C = \sum_{i=1}^{n} \#CTR_i(V_{i1},..,V_{ik_i})$[1]. Throughout this paper we consider a more general case of the *cardinality* operator where we associate to each constraint CTR_i $(1 \le i \le n)$ a weight $W_i \in \mathbb{N}$; the *weighted cardinality* operator

[1] As usual within constraint programming, this definition applies for the ground case when all the variables of the constraints $CTR_1,..,CTR_n$ are fixed.

holds iff $C = \sum_{i=1}^{n} \left(W_i \cdot \#CTR_i\left(V_{i1},..,V_{ik_i}\right)\right)$. This extension is for instance useful for those problems where the constraints have different weights and where the objective is to maximize the sum of the weights of the constraints that hold.

From an operational point of view the *cardinality* operator used entailment [4] in order to implement the corresponding propagation. However a fundamental weakness is that it assumes each constraint to be independent. This is not the case in practice since, very often, the same variable occurs in more than one constraint. The contribution of this paper is to propose new necessary conditions that take advantage of the fact that some variables occur in more than one constraint. In addition it also discusses possible ways to use them for pruning variables.

2 Necessary Conditions for the *weighted cardinality* Operator

Consider any partitioning of the collection of constraints $\left\{CTR_1\left(V_{11},..,V_{1k_1}\right)..,CTR_n\left(V_{n1},..,V_{nk_n}\right)\right\}$ of the *weighted cardinality* operator into p $(1 \le p \le n)$ conjunctions of constraints, where C_i $(1 \le i \le p)$ denotes the set of indices of those constraints that belong to the i^{th} conjunction, and $|C_i|$ the number of elements of C_i.

Theorem 1. *The weighted cardinality operator can be satisfied only if :*

$$\min(C) \le \sum_{i=1}^{n} W_i - \sum_{i=1}^{p} \left(\min_{j \in C_i} \left(W_j \right) \cdot \left(1 - \#\left(\bigwedge_{j \in C_i} CTR_j\left(V_{j1},..,V_{jk_j} \right) \right) \right) \right) . \qquad (1)$$

Proof. For each conjunction of constraints such that we find a contradiction, we have that at least one constraint can't be satisfied, so we compute the minimum weight of the constraints of that conjunction. Since no constraint occurs in more than one conjunction, and by adding the previous minimum weights, we get a lower bound of the sum of the weight of constraints that do not hold. By subtracting it from the sum of the weights of all constraints, we obtain an upper bound of the sum of the weights of the constraints that hold. Since C should not be greater than this upper bound, a necessary condition is as follows: the minimum value of C should be less than or equal to this upper bound. ❑

Theorem 1 allows adjusting the maximum value of C. The conjunctions of constraints can be obtained e.g. by using the following heuristic which tries to extend an existing conjunction as long as contradiction is not reached: in order to reach contradiction more rapidly the next constraint to incorporate into a conjunction should share as many variables as possible with the variables that already occur in the constraints of that conjunction.

As a corollary of Theorem 1 we have Theorem 2 which allows enforcing a conjunction of constraints.

Theorem 2. *The weighted cardinality constraint can be satisfied only if each conjunction of constraints* h $(1 \le h \le p)$*, such that condition* (2) *is true, holds.*

$$\min(C) > \sum_{i=1}^{n} W_i - \min_{i \in C_h}(W_i) - \sum_{i=1, i \neq h}^{p} \left(\min_{j \in C_i} \left(W_j \right) \cdot \left(1 - \# \left(\bigwedge_{j \in C_i} CTR_j \left(V_{j1}, .., V_{jk_j} \right) \right) \right) \right) \quad (2)$$

Theorem 2 can be interpreted as follows: if C is for sure greater than the upper bound obtained by considering all constraints except the one that corresponds to the minimum weight in conjunction h, then each constraint of conjunction h should be enforced.

Finally we explain how to prune the domain of a variable V which occurs in at least one constraint of the *weighted cardinality* operator.

Theorem 3. *The weighted cardinality operator can be satisfied only if condition* (3) *or* (4) *holds, for every* V *and* $val \in \mathrm{dom}(V)$.

$$V \neq val \quad (3)$$

$$\min(C) \le \sum_{i=1}^{n} W_i - \sum_{i=1}^{p} \left(\min_{j \in C_i} \left(W_j \right) \cdot \left(1 - \# \left(\bigwedge_{j \in C_i} CTR_j \left(V_{j1}, .., V_{jk_j} \right) \wedge V = val \right) \right) \right) \quad (4)$$

Proof. The right hand side of condition (4) corresponds to an upper bound of the sum of the weights of the constraints that hold for any conjunction of constraints where $V = val$. If C can't be less than or equal to this upper bound, then the only way to avoid violating the *weighted cardinality* operator is to remove value val from $\mathrm{dom}(V)$. ❑

A first possible heuristic for using Theorem 3 for pruning a variable V is as follows: partition the constraints occurring in the *weighted cardinality* operator in such a way that each constraint mentioning variable V is put in a singleton conjunction. Let denote $CTR_{V1}, .., CTR_{Vm}$ the constraints where V occurs and let $\mathrm{dom}(V|CTR_i)$ $(1 \le i \le n)$ denote the domain of variable V under the assumption that constraint CTR_i is enforced (it is empty if constraint CTR_i leads to a contradiction). Furthermore let d be the sum of the weights of those constraints CTR_i $(i \in \{V1, .., Vm\})$. Thus, from Theorem 3 we derive the following rule: If $\min(C) > \sum_{i=1}^{n} W_i - \sum_{i \in 1..p, C_i \cap \{V1, .., Vm\} = \varnothing} \left(\min_{j \in C_i} (W_j) \cdot \left(1 - \# \left(\bigwedge_{j \in C_i} CTR_j \left(V_{j1}, .., V_{jk_j} \right) \right) \right) \right) - d$ then all values $val \notin \bigcup_{i \in \{V1, .., Vm\}} \mathrm{dom}(V|CTR_i)$ should be removed from $\mathrm{dom}(V)$.

Theorem 3 can also be used for adjusting the minimum value of a variable V occurring in the *cardinality* operator (i.e. we assume all weights to be equal to 1). Let $\min(V|CT)$ denotes the minimum of variable V under the assumption that the constraint or the conjunction of constraints CT is enforced (it is equal to $\max(V)+1$ if

CT leads to a contradiction). Furthermore, let us define the k^{th} smallest value of a multiset of natural numbers $X_1,..,X_l$ $(l \geq k)$ as the smallest value X_i $(1 \leq i \leq l)$ such that there exist at least $k-1$ X_j $(j \neq i, 1 \leq j \leq l)$ that are less than or equal to X_i. Again from Theorem 3 we get the following pruning rule:

If $\min(C) > n - p$ then we can adjust the minimum value of V to the $(\min(C) - n + p)^{th}$ smallest value among values $\min\left(V \mid \bigwedge_{j \in C_i} CTR_j\left(V_{j1},..,V_{jk_j}\right)\right)$ $(1 \leq i \leq p)$.

We have a similar rule for adjusting the maximum value of variable V.

Three theorems similar to Theorems 1, 2, 3 can also be obtained by considering conjunctions of negation of constraints. The pruning rules of the *cardinality* operator provided in [2] can be derived from Theorems 1 and 2 by setting the number of constraints in a conjunction, as well as all the weights, to 1. Constructive disjunction [1], [4] can also be obtained from Theorem 3 by associating each alternative of the disjunction to a singleton conjunction, by setting C to 1, and by setting all the weights to 1. For a specific constraint network, how to find out the way to partition the constraints of the *weighted cardinality* operator that leads to the maximum amount of pruning, is a natural question that arises from this paper.

Acknowledgments

Thanks to Per Kreuger, Per Mildner and Emmanuel Poder for helpful comments and suggestions on this paper.

References

1. Würtz, J., Müller, T.: Constructive Disjunction Revisited. In *20th German Annual Conference on Artificial Intelligence*. LNAI vol. 1137, 377-386, Springer-Verlag, (1996).
2. Van Hentenryck, P., Deville, Y.: The Cardinality Operator: A New Logical Connective for Constraint Logic Programming. In *International Conference on Logic Programming*. The MIT Press, 745-759, (1991).
3. Van Hentenryck, P.: *Constraint Satisfaction in Logic Programming*. The MIT Press, (1989).
4. Van Hentenryck, P., Saraswat, V., Deville, Y.: Design, Implementation and Evaluation of the Constraint Language cc(*FD*). In *A. Podelski, ed., Constraints: Basics and Trends*, vol. 910 of Lecture Notes in Computer Science, Springer-Verlag, (1995).