Constraints
DOI 10.1007/s10601-012-9138-7

Multiset variable representations and constraint propagation

Y. C. Law · J. H. M. Lee · T. Walsh · M. H. C. Woo

Abstract Multisets generalize sets by allowing elements to have repetitions. In this paper, we study from a formal perspective representations of multiset variables, and the consistency and propagation of constraints involving multiset variables. These help us model problems more naturally and can, for example, prevent introducing unnecessary symmetries into a model. We identify a number of different representations for multiset variables, compare them in terms of effectiveness and efficiency, and propose inference rules to enforce bounds consistency for the representations. In addition, we propose to exploit the variety of a multiset—the number of distinct elements in it—to improve modeling expressiveness and further enhance constraint propagation. We derive a number of inference rules involving the varieties of multiset variables. The rules interact varieties with the traditional components of multiset variables (such as cardinalities) to obtain stronger propagation. We also demonstrate how to apply the rules to perform variety reasoning on some common multiset constraints. Experimental results show that performing variety reasoning on top of cardinality reasoning can effectively reduce more search space and achieve better runtime in solving some multiset CSPs.

Keywords Constraint satisfaction · Multiset variables

Y. C. Law · J. H. M. Lee · M. H. C. Woo (✉)
Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
e-mail: hcwoo@cse.cuhk.edu.hk

Y. C. Law
e-mail: yclaw@cse.cuhk.edu.hk

J. H. M. Lee
e-mail: jlee@cse.cuhk.edu.hk

T. Walsh
NICTA & UNSW, Sydney, Australia
e-mail: toby.walsh@nicta.com.au

Published online: 26 February 2013

1 Introduction

Many combinatorial design problems can be modeled as *constraint satisfaction problems* (CSPs) [16] using *set variables*, which take collections of distinct elements as their values. The domain of a set variable is typically represented by its set upper and lower bounds, and propagated by enforcing set bounds consistency [8] together with cardinality reasoning [1, 2]. Utilizing also the cardinality information of a set variable during propagation allows for more prunings than using set bounds propagation alone, and further reduces the search space.

Set variables have been incorporated into most of the major constraint solvers (see, for example, Conjunto [7], ILOG [11], and Mozart [18]). Similar to sets, multisets are also unordered collections of items but allow repetitions of elements. Given the success of set variables and constraints in providing more natural and efficient models to many problems and avoiding unnecessary symmetries in the models, it is surprising to find little work on multiset constraint solving. Whilst multiset variables are supported in a few solvers (for example, ILOG's configurator supports multiset variables), there has been little work from a more theoretical perspective on propagating such variables. The aim of this paper is to *rectify this imbalance, to study formal notions of consistency and propagation for multiset variables, and to discuss how they can be implemented.* Many problems naturally involve multisets. Consider the template design problem (prob002 in CSPLib [6]) in which we assign designs to printing templates. As each template contains a fixed number of slots, we can model this problem with an integer variable for each slot, whose value is the design in this slot. However, this model introduces unnecessary symmetries as the slots are indistinguishable. Since a design can appear multiple times in one template, a more natural model is to use a multiset variable for each template to avoid the symmetries. The domain of each variable is the set of all possible multisets of designs that can be assigned to the template.

This paper combines and extends the work of Walsh [22] and Law et al. [15]. We first introduce three different representations, namely bounds, occurrence, and fixed cardinality representations [22], for multiset variables and compare the tightness of bounds among the representations. In a CSP model, the constraints often have a mixture of multiset, set, and/or integer variables. We propose various simple inference rules which enforce bounds consistency for constraints involving these kinds of variables. We also consider the *variety* of a multiset variable, which is the number of *distinct* elements [15]. Based on the occurrence representation, we further propose a hybrid representation [15] for multiset variables by incorporating a cardinality variable as well as a variety variable. The cardinality of a set reveals the *total* number of elements in it. Incorporating a cardinality variable to a set variable [1, 2] enjoys success in enhancing propagation for set constraints. We hope to benefit from incorporating cardinality and variety variables to multiset variables as well. Our hybrid representation not only allows us to express certain problem constraints much more easily (i.e., better modeling expressiveness), but also increases the opportunities to infer more domain prunings for better solving efficiency. We derive a number of inference rules involving the varieties of multiset variables and show how the traditional components of multiset variables (such as cardinalities) interact with the varieties to achieve stronger constraint propagation. We also apply our rules to perform variety reasoning on some common multiset constraints. Experimental

results confirm that performing variety reasoning on top of cardinality reasoning can further reduce the search space and give a better runtime in solving CSPs involving multiset variables.

The rest of the paper is organized as follows. Section 2 describes set variables and how set constraints can be used in modeling combinatorial problems. Section 3 presents a number of multiset representations, some common multiset constraints, and the local consistency for constraint propagation. Section 4 focuses on how cardinality reasoning helps to improve expressivity and constraint propagation in multiset constraints. Section 5 introduces the variety information in multiset variables and shows how inferences using variety, together with cardinality, can achieve stronger constraint propagation. Section 6 presents the experimental results on some multiset problems. Section 7 summarizes the contributions of this paper and sheds light on possible directions of future research.

2 Set variables and constraint propagation

In this section, we define constraint satisfaction problems (CSPs) formally. Since our work on multiset variables is directly related to that of set variables, it is also useful to review the relevant definitions and concepts.

2.1 Constraint satisfaction problems

A *constraint satisfaction problem* (CSP) is a triple $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a finite set of *variables*, $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of finite *domains* of possible values for each variable, and $\mathcal{C}$ is a set of *constraints*. Each constraint involves a subset of variables in $\mathcal{X}$, limiting the combination of values that the variables in the subset can take. An *assignment* $x_i \mapsto a$ in $\mathcal{P}$ is a mapping from variable x_i to value $a \in D_{x_i}$. A *solution* of $\mathcal{P}$ is a set of assignments of all variables in $\mathcal{P}$ that satisfies all the constraints in $\mathcal{C}$.

2.2 Set constraint satisfaction

A *set* is an unordered collection of elements *without repetitions*. The *cardinality* of a set S is the number of elements in S, denoted as $|S|$. Given a universe U of integers $\{1, \ldots, n\}$, a set variable S takes its set values from U. Sets are denoted by the letters m, M, s, t, x, and y. A set value m of cardinality c is denoted by $\{m_1, m_2, \ldots, m_c\}$ where $m_1 < m_2 < \cdots < m_c$ and m_j denotes the j-th smallest value in m.

Gervet [8] proposed the *bounds representation* of the domain of a set variable S with an interval $[glb(S), lub(S)]$ such that $D_S = \{m \mid glb(S) \subseteq m \subseteq lub(S)\}$. The *greatest lower bound* $glb(S)$ contains all the elements which *must exist* in the set, while the *least upper bound* $lub(S)$ contains any element which *may exist* in the set. S is said to be *bound* when its lower bound equals its upper bound (i.e., $glb(S) = lub(S)$). Figure 1 gives a lattice which represents a set domain of a set variable S with an interval $[\emptyset, \{1, 2, 3, 4\}]$. Each edge in the figure represents a subset relation in which the set value below is a subset of the set value above on the two ends of the edge.

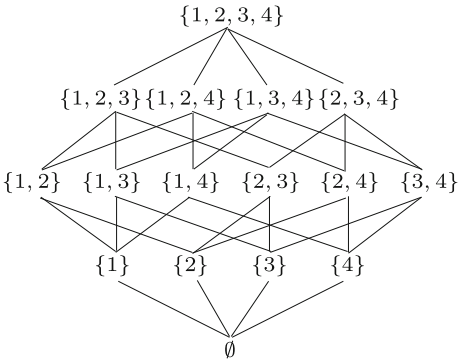

Fig. 1 The domain of a set variable S with interval $[\emptyset, \{1, 2, 3, 4\}]$

Example 2.1 Steiner triple system (prob044 in CSPLib)
The Steiner triple problem of order n consists of finding a set of $n(n-1)/6$ triples of distinct integer elements in $\{1, \ldots, n\}$ such that any two triples have at most one common element. To model this problem, we use two kinds of set variables. The first kind of variables S_i, where $i \in \{1, \ldots, n(n-1)/6\}$, denotes triples in the problem. The other kind of variables A_{ij}, $\forall i, j \in \{1, \ldots, n(n-1)/6\}$ and $i < j$, denotes the intersection of any two triples S_i and S_j. The domains of the two kinds of variables are both $D_{S_i} = D_{A_{ij}} = [\emptyset, \{1, \ldots, n\}]$. The constraints of this problem are as follows:

- intersection: $S_i \cap S_j = A_{ij}$, and
- cardinality: $|S_i| = 3$, $|A_{ij}| \leq 1$,

for all $i, j \in \{1, \ldots, n(n-1)/6\}$ and $i < j$.

The Steiner triple problem of order 7 has $7 \times (7-1)/6 = 7$ triples. One possible solution is $\{1, 2, 3\}$, $\{1, 4, 5\}$, $\{1, 6, 7\}$, $\{2, 4, 6\}$, $\{2, 5, 7\}$, $\{3, 4, 7\}$, $\{3, 5, 6\}$.

2.2.1 Tree searching

In solving a CSP with set variables, depth-first backtracking search [20] is often used. At each search node, the algorithm first chooses an unbound set variable S. Instead of assigning a value from the domain of the selected variable, the algorithm selects an element m that exists in the upper bound but not in the lower bound (i.e., $m \in (lub(S) \setminus glb(S))$) of the domain of S and proceeds the search with two branches: $m \in S$ and $m \notin S$. Thus, at each node, the algorithm splits the search space into two. When we set $m \notin S$, the element m is removed from $lub(S)$ of the domain of S. When we set $m \in S$, the element m is added to $glb(S)$ of the domain of S.

Example 2.2 Consider the set variable S in Fig. 1, where $D_S = [\emptyset, \{1, 2, 3, 4\}]$. When we set $1 \notin S$, the element 1 is removed from $lub(S)$ and the domain becomes $[\emptyset, \{2, 3, 4\}]$. Figure 2 gives the modified domain of S in which all the sets that contain element 1 are removed from the lattice.

2.2.2 Set bounds consistency

Traditional domain reasoning, such as generalized arc consistency [17], for integer variables is not practical for set variables, as their domains are exponential in the size

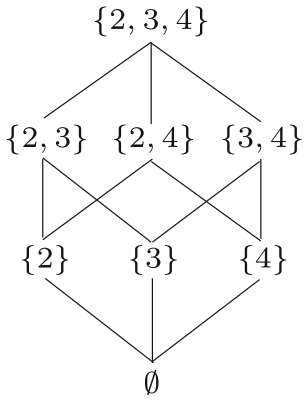

Fig. 2 The modified set domain of S

of possible sets. Gervet [8] proposed using bounds reasoning to maintain consistency on set variables.

A set variable S with interval domain $[glb(S), lub(S)]$ is *set bounds consistent* [8] with respect to a constraint C if and only if $glb(S) = \cap dom_S(C)$ and $lub(S) = \cup dom_S(C)$, where $dom_S(C)$ denotes the domain values of S that satisfy C.

Set bounds consistency can be enforced by *projection functions* [8]. Each set constraint is associated with a set of projection functions which derive new bounds for the set domains to maintain set bounds consistency. Table 1 shows the projection functions of some common set constraints.

Example 2.3 Suppose we have set variables S_1 and S_2, where $D_{S_1} = [\{1, 2\}, \{1, 2, 3, 4\}]$ and $D_{S_2} = [\emptyset, \{1, 2, 3\}]$, and a constraint $S_1 \subseteq S_2$. Both variables are not set bounds consistent because the element $4 \in S_1$ does not exist in S_2 and the elements 1 and 2 are not yet included in $glb(S_2)$. After enforcing set bounds consistency, we have $D_{S_1} = D_{S_2} = [glb(S_1) \cup glb(S_2), lub(S_1) \cap lub(S_2)] = [\{1, 2\}, \{1, 2, 3\}]$. The variables S_1 and S_2 become set bounds consistent with respect to the subset constraint.

2.3 Cardinality variable and reasoning

Conjunto [8] is one of the first constraint solvers developed in which a set variable is represented by set intervals. Other set constraint solvers include Oz [19], Choco [13],

Table 1 Projection functions of some common set constraints

Subset	$lub(S_1) \leftarrow lub(S_1) \cap lub(S_2)$
$(S_1 \subseteq S_2)$	$glb(S_2) \leftarrow glb(S_2) \cup glb(S_1)$
Union	$glb(S_1) \leftarrow glb(S_1) \cup glb(S_3) \setminus lub(S_2)$
$(S_1 \cup S_2 = S_3)$	$lub(S_1) \leftarrow lub(S_1) \cap lub(S_3)$
	$glb(S_3) \leftarrow glb(S_3) \cup glb(S_1) \cup glb(S_2)$
	$lub(S_3) \leftarrow lub(S_3) \cap lub(S_1) \cup lub(S_2)$
Intersection	$glb(S_1) \leftarrow glb(S_1) \cup glb(S_3)$
$(S_1 \cap S_2 = S_3)$	$lub(S_1) \leftarrow lub(S_1) \setminus ((lub(S_1) \cap glb(S_2)) \setminus lub(S_3))$
	$glb(S_3) \leftarrow glb(S_3) \cup glb(S_1) \cap glb(S_2)$
	$lub(S_3) \leftarrow lub(S_3) \cap lub(S_1) \cap lub(S_2)$

Mozart [18], the ROBDD-based [10, 14], and Gecode [5]. Azevedo and Barahona [1, 2] further proposed *cardinality reasoning* on set constraints and developed another set constraint solver, *Cardinal*, to handle the set cardinality more actively and improve the performance in solving CSPs with set variables.

On the other hand, Gervet and Van Hentenryck [9] proposed representing set variables using length-lex bound. This representation totally orders a set domain, and incorporates the cardinality and the position in lexicographic ordering directly. Whilst this representation increases propagation, it comes at the expense of potentially exponential cost. Even reaching a fixed point with bounds reasoning may require an exponential number of iterations. In this paper, by comparison, we consider representations for sets and multisets which are guaranteed to take only a polynomial number of iterations but nevertheless increase the amount of propagation.

3 Multiset constraint satisfaction

A *multiset* is a generalization of set that allows elements to repeat. Without loss of generality, we assume that multiset elements are positive integers from 1 to n. We shall use $\emptyset$ to denote both the empty set and the empty multiset. The universe of a multiset is a multiset itself, which defines the maximum possible occurrences of each element. Given a universe U, we denote a multiset by $S = \{\!\{m_1, m_2, \cdots, m_c\}\!\} \subseteq U$ where $m_i \leq m_j$ for $1 \leq i \leq j \leq c$, and its cardinality (total number of elements) as $|S|$. Multisets are denoted by letters s, t, x, and y. For example, if $S = \{\!\{1, 1, 2, 2, 3\}\!\}$, then $|S| = 5$. Since an element in a multiset variable can occur multiple times, we let $occ(i, S)$ be the number of occurrences of an element i in the multiset S.

3.1 Basic multiset variable representations

A multiset variable allows an element to repeat. It is not practical to represent a multiset variable as a finite domain variable in which the domain is a set of all possible multisets. The number of possible multisets grows exponentially as the number of elements in a multiset variable increases. Thus, we suggest two basic multiset representations: bounds and occurrence representations.

3.1.1 Bounds representation

The bounds representation for multiset variables is a generalization of the bounds representation for set variables [8]. The domain of a multiset variable S is specified by an interval $[glb(S), lub(S)]$. The *greatest lower bound* $glb(S)$ is the largest multiset containing all the values that *must exist* in the multiset. The *least upper bound* $lub(S)$ is the smallest multiset containing all the values that *may exist* in the multiset. This representation is compact yet cannot represent all forms of disjunction because it may contain unnecessary values between the upper and lower bounds.

Example 3.1 Consider a multiset variable S with two possible multiset values: $\{\!\{1\}\!\}$ and $\{\!\{2, 2\}\!\}$. Based on the bounds representation, S has a domain $D_S = [\emptyset, \{\!\{1, 2, 2\}\!\}]$. However, this representation also permits S to take the values $\emptyset$, $\{\!\{2\}\!\}$, $\{\!\{1, 2\}\!\}$, and $\{\!\{1, 2, 2\}\!\}$ which are not possible values of S.

3.1.2 Occurrence representation [22]

In the occurrence representation, a multiset variable with n elements is represented by a vector $\langle x_1, \ldots, x_n \rangle$ of integer variables $x_i = occ(i, S)$. This vector is known as the *occurrence vector*. This representation is compact but, similar to the bounds representation, cannot represent all forms of disjunction.

Example 3.2 Consider a multiset variable S with two possible multiset values: $\{\{1\}\}$ and $\{\{2, 2\}\}$. Based on the occurrence representation, S has an occurrence vector $\langle x_1, x_2 \rangle$, where $D_{x_1} = D_{occ(1,S)} = \{0, 1\}$ and $D_{x_2} = D_{occ(2,S)} = \{0, 2\}$. However, this representation also permits S to take the values $\emptyset$ and $\{\{1, 2, 2\}\}$ which are not possible values of S.

3.1.3 Expressivity

Based on the two different representations, we compare their expressivities. Given A and B be two different multiset representation methods. A is said to be *as expressive as* B if A and B both can represent the same set of multisets. A is said to be *more expressive* than B if (1) A is as expressive as B, and (2) there exists a set of multisets in which A can represent them with tighter bounds than B. A and B are *incomparable* if neither one of them is as expressive as the other.

Note that the occurrence representation gives the same representation as the bounds representation if we only maintain the bounds (i.e., the lower and upper bounds of the interval domain) of the variables in the occurrence vector.

Theorem 3.1 *The occurrence representation is as expressive as the bounds representation if the occurrence representation is restricted to maintain only bounds on the number of occurrences of an element in a multiset.*

Proof Suppose we have some bounds representation. We construct an occurrence representation containing all the integers between the upper and lower bounds in the bound representation. This represents precisely the same multisets as the bounds representation. □

In general, however, the occurrence representation is more expressive than the bounds representation.

Theorem 3.2 *The occurrence representation is more expressive than the bounds representation.*

Proof By Theorem 3.1, the occurrence representation is as expressive as the bounds. To show that the occurrence representation is more expressive, we give an example where the occurrence representation is tighter. Consider a multiset variable S with two values: $\emptyset$ or $\{\{1, 1\}\}$. This can be represented exactly with the occurrence variable $D_{x_1} = occ(1, S) = \{0, 2\}$. By comparison, a bounds representation would need $glb(S) = \emptyset$ and $lub(S) = \{\{1, 1\}\}$, and this permits the additional value $\{\{1\}\}$. □

In the rest of this paper, we mainly focus on the occurrence representation for multiset variables. A multiset variable S is represented by a vector

$\langle occ(1, S), \ldots, occ(n, S)\rangle$, denoting the number of occurrences of each element in S. The domain of S is denoted as the interval $D_{occ(i,S)} = [occ_r(i, S), occ_p(i, S)]$ in which $occ_r(i, S)$ and $occ_p(i, S)$ refer to the lower and upper bound of the number of occurrences of each element i respectively. We also define s_r and s_p as the multisets whose number of occurrences of each element i is $occ_r(i, S)$ and $occ_p(i, S)$ respectively. The multisets s_r and s_p are in fact the greatest lower bound $glb(S)$ and the least upper bound $lub(S)$ of S respectively (i.e., $D_S = [glb(S), lub(S)] = [s_r, s_p] = [\langle occ_r(1, S), \ldots, occ_r(n, S)\rangle, \langle occ_p(1, S), \ldots, occ_p(n, S)\rangle]$.)

A set value can also be represented using the occurrence representation in which the number of occurrence is either 0 or 1 to denote the existence of the corresponding element.

3.2 Multiset bounds consistency

Gervet [8] proposed using bounds reasoning to maintain consistency on set variables due to the exponential size of set domains. Similarly, we consider using bounds consistency (BC) on the occurrence representation of multiset variables. We propose a new definition of local consistency that works with constraints involving multiset variables. Given a constraint C over the variables $X_1, \ldots, X_n$ and $sol(X_i)$ represents the values for X_i which can be extended to the other variables. That is, $sol(X_i) = \{D_{X_i} \mid C(D_{X_1}, \ldots, D_{X_n}) \wedge \forall j \,.\, glb(X_j) \subseteq D_{X_j} \subseteq lub(X_j)\}$.

A constraint $C(X_1, \ldots, X_n)$ is *BC* if and only if for each multiset variable X_j in the constraint, $sol(X_j) \neq \emptyset$, and $glb(X_j) = \bigcap_{m \in sol(X_j)} m$ and $lub(X_j) = \bigcup_{m \in sol(X_j)} m$. This definition of local consistency might look rather expensive, being defined over the set of *all* solutions. However, this set merely identifies *support* for particular values in the set or multiset.

When using BC to filter, we will identify values which occur in no solution and so can be pruned. Thus, we will not be finding all solutions but merely identifying those values that occur in no solution (i.e., lack support). We define a bound support to be a satisfying assignment M of a constraint in which for each multiset/set variable X, $glb(X) \subseteq M \subseteq lub(X)$. The following theorem justifies why BC can be called "bounds consistency".

Theorem 3.3 *If we enforce BC on the occurrence representation of multiset and/or set variables, then for each element m in any multiset/set variable X, both* $occ(m, glb(X))$ *and* $occ(m, lub(X))$ *have bound supports.*

Proof Suppose that a constraint is BC. Consider any multiset/set variable X in the constraint. We can construct an equivalent occurrence representation. Suppose $m_{\max} = occ(m, lub(X))$ and $m_{\min} = occ(m, glb(X))$. Then we let the variable X_m in the occurrence vector have a domain $[m_{\min}, m_{\max}]$. Consider $X_m = m_{max}$. From the definition of BC and the generalized multiset union operator, there must be a satisfying solution to the constraint in which $occ(m, X) = m_{\max}$. If there are several, we choose one non-deterministically. Similarly, there must be a satisfying solution to the constraint when we consider $X_m = m_{\min}$. Hence, the result holds. □

Unfortunately, the occurrence representation increases the number of variables in the problem. For example, suppose we have a constraint like $X \neq Y$ where

X and Y are multiset variables. This maps into a large disjunctive constraint in the occurrence representation over $2d$ integer variables where d is the maximum possible cardinality of the two multisets. It is therefore worth developing specialized propagation algorithms that exploit the semantics of set or multiset constraints. Such algorithms can work on either a bounds or an occurrence representation.

In the next two subsections, we show how to define such algorithms by means of some simple inference rules. Note that a degenerate version of this last theorem is that BC on a constraint containing just integer variables is equivalent to bounds consistency on these variables. Some other properties also follow immediately from this result.

3.3 Multiset constraints

Most set constraints can be generalized to their multiset counterparts. Table 2 gives some common multiset constraints, in which X, Y, and Z are multiset variables and i is an element. Note that the union multiset constraint takes the maximum number of occurrences of each element in X and Y, while the union-plus multiset constraint sums up the number of occurrences of each element in X and Y.

A multiset expression is, in turn, a ground multiset, a multiset variable, or an expression of the form $X \cup Y$, $X \uplus Y$, or $X \cap Y$ where X and Y are again multiset expressions. To make constraint propagation easier, we decompose constraints into a flattened normal form in which constraints are at most ternary and only of the form: $X = Y$, $X \subseteq Y$, $X \cup Y = Z$, $X \uplus Y = Z$, $X \cap Y = Z$, $|X| = N$, $occ(m, X) = N$ where X, Y, and Z are multiset variables or ground multisets, N is an integer variable or an integer constant. This *normalization* takes any nested multiset expression and replaces it by a new equality constraint. For example, $(X \cup Y) \subseteq Z$ is normalized to give $XY = X \cup Y$ and $XY \subseteq Z$ where XY is a new multiset variable. A similar decomposition of set constraints is used in [8]. In general, such decomposition hinders constraint propagation.

Following Debruyne and Bessière, we define the notion of *stronger* between two local consistencies A and B. A is *stronger* [4] than B ($A \geq B$) if in any CSP in which A holds, then B holds too.

Theorem 3.4 *BC on a set of constraints is stronger than BC on the equivalent set of constraints decomposed into normal form.*

Proof Clearly it is as strong since normalization merely replaces one constraint with a set of logically equivalent constraints. For strictness, we can consider any type of multiset or set constraint. For example, for the set not-equals constraint, consider $X \cup (Y \cap Z) \neq (X \cup Y) \cap (X \cup Z)$ with $D_X = D_Y = D_Z = [\emptyset, \{0\}]$. BC

Table 2 Some common multiset constraints

Equality	$X = Y$ iff $occ(i, X) = occ(i, Y), \forall i \in X, Y$
Subset	$X \subseteq Y$ iff $occ(i, X) \leq occ(i, Y), \forall i \in X, Y$
Union	$X \cup Y = Z$ iff $occ(i, Z) = \max(occ(i, X), occ(i, Y)), \forall i \in X, Y, Z$
Union-Plus	$X \uplus Y = Z$ iff $occ(i, Z) = occ(i, X) + occ(i, Y), \forall i \in X, Y, Z$
Intersection	$X \cap Y = Z$ iff $occ(i, Z) = \min(occ(i, X), occ(i, Y)), \forall i \in X, Y, Z$

determines that this constraint has no solution. But in the decomposition, with $YZ = Y \cap Z$, $XYZ = X \cup YZ$, $XY = X \cup Y$, $XZ = X \cup Z$, $XYXZ = XY \cap XZ$, and $XYZ \neq XYXZ$, the domains $D_X = D_Y = D_Z = D_{YZ} = D_{XYZ} = D_{XY} = D_{XZ} = D_{XYXZ} = [\emptyset, \{0\}]$ make the decomposed constraints BC. Similar arguments hold for the other types of constraints. Note that for strictness, we do not need to give such arguments as it is sufficient to identify just one constraint (in this case, set not-equals) on which decomposition into the normal form hinders pruning. □

Using the simple restriction that there are no repeated occurrences of variables on the right-hand side of every constraint, decomposition does not hinder constraint propagation.

Theorem 3.5 *BC on a set of constraints, none of which contains a repeated occurrence of variables, is equivalent to BC on the equivalent set of constraints decomposed into normal form.*

Proof (Outline) The proof uses induction on the number of auxiliary variables introduced and the structure of the multiset expressions which they replace, followed by extensive case analysis. The step cases of this proof reduce to consider each possible ternary constraint on set or multiset variables and demonstrating that BC on normalisation of such a constraint does not hinder BC reasoning. Consider, for example, the multiset constraint $X \cup Y \subseteq Z$ and the decomposition: $XY = X \cup Y$, $XY \subseteq Z$. Suppose each of the decomposed constraints is BC but the original undecomposed constraint is not BC. There are four cases. In the first case, $glb(Z)$ is too small and we can add at least one value m to it. This is only possible if m is in $glb(X)$ or in $glb(Y)$. But then m is in $glb(XY)$ and thus in $glb(Z)$, which is a contradiction. In the second case, $lub(X)$ is too large and we can delete at least one value m from it and m is also not in $lub(Y)$. But then m is not in $lub(XY)$ and thus not in $lub(Z)$, which is a contradiction. In the third case, $lub(Y)$ is too large and we can delete at least one value m from it and m is also not in $lub(X)$. But then m is not in $lub(XY)$ and thus not in $lub(Z)$, which is a contradiction. In the fourth case, $lub(X)$ and $lub(Y)$ are both too large and we can delete at least one value m from both of them. But then m is not in $lub(XY)$ and thus not in $lub(Z)$, which is a contradiction. This covers all possible ways in which the bounds on X, Y, and Z are pruned. For instance, $lub(Z)$ is never pruned as the subset constraint only constrains $glb(Z)$. The analysis for other constraints is similar. Consider, for example, the multiset constraint $X \cap Y \subseteq Z$ and the decomposition: $XY = X \cap Y$, $XY \subseteq Z$. There are three cases. In the first case, $glb(Z)$ is too small and we can add at least one value m to it. This is only possible if m is in $glb(X)$ and in $glb(Y)$. But then m is in $glb(XY)$ and thus in $glb(Z)$, which is a contradiction. In the second case, $lub(X)$ is too large and we can delete at least one value m from it where m is in $glb(Y)$. But then m is not in $lub(XY)$ and thus not in $lub(Z)$, which is a contradiction. In the third case, $lub(Y)$ is too large and we can delete at least one value m from it where m is in $glb(X)$. But then m is not in $lub(XY)$ and thus not in $lub(Z)$, which is a contradiction. This again covers all possible ways in which the bounds on X, Y, and Z are pruned. □

3.4 Enforcing local consistency

We now give some simple constraint propagation rules that enforce BC on multiset constraints in normal form. Following Azevedo [1, 2] and Walsh [22], inference rules will be formally described as rewriting rules as in the following schematic figure:

$$\text{(trigger condition)} \quad \frac{\text{conditions (which can be nil)}}{\text{changes in constraint store}}$$

Note that X, Y, and Z are multiset variables or ground multisets and N is an integer variable or an integer constant. The changes in constraint store will be in the form of {old constraints} $\mapsto$ {new constraints} where the original set of constraints (old constraints) will be replaced by another equivalent set of constraints (new constraints).

Equality constraint ($X = Y$)

When X and Y are forced to be equal, both X and Y contain the same number of occurrences of every element i.

$$\frac{}{\{X = Y\} \mapsto \{occ(i, X) = occ(i, Y)\}} \tag{1}$$

Subset constraint ($X \subseteq Y$)

When Y contains X, the number of occurrences of each element i in Y is either greater than or equal to those in X.

$$\frac{}{\{X \subseteq Y\} \mapsto \{occ(i, X) \leq occ(i, Y)\}} \tag{2}$$

Union constraint ($X \cup Y = Z$)

Union takes the maximum number of occurrences of each element between X and Y. When Z is the union of X and Y, $occ(i, Z) = \max(occ(i, X), occ(i, Y))$ for all elements i.

$$\frac{}{\begin{array}{l}\{X \cup Y = Z\} \mapsto \{occ(i, Z) = \max(occ(i, X), occ(i, Y)),\\ \qquad\qquad occ(i, X) \leq occ(i, Z), occ(i, Y) \leq occ(i, Z)\}\end{array}} \tag{3}$$

Union-plus constraint ($X \uplus Y = Z$)

Union-plus sums up all the elements in both X and Y. When Z is the union-plus of X and Y, $occ(i, Z) = occ(i, X) + occ(i, Y)$ for all elements i.

$$\frac{}{\begin{array}{l}\{X \uplus Y = Z\} \mapsto \{occ(i, Z) = occ(i, X) + occ(i, Y),\\ \qquad\qquad occ(i, X) \leq occ(i, Z), occ(i, Y) \leq occ(i, Z)\}\end{array}} \tag{4}$$

Intersection constraint ($X \cap Y = Z$)

Intersection takes the minimum number of occurrences of each element between X and Y. When Z is the intersection of X and Y, $occ(i, Z) = \min(occ(i, X), occ(i, Y))$

for all elements i.

$$\frac{}{\{X \cap Y = Z\} \mapsto \{occ(i, Z) = \min(occ(i, X), occ(i, Y)), \; occ(i, Z) \leq occ(i, X), occ(i, Z) \leq occ(i, Y)\}} \quad (5)$$

Cardinality constraint ($|X| = N$)

Cardinality of X refers to the total number of elements in X, which is the sum of the number of occurrences of each element i.

$$\frac{}{\{|X| = N\} \mapsto \{\textstyle\sum_i occ(i, X) = N\}} \quad (6)$$

Failure rules

A failure can be detected when the lower bound x_r of X is not included in the upper bound x_p of X.

$$(X \text{ changed bounds}) \quad \frac{\text{not}(x_r \subseteq x_p)}{\{\} \mapsto \text{fail}} \quad (7)$$

Each rule tightens an upper and/or lower bound on a variable. The rules therefore terminate either with domains at a fixed point or by flagging failure. The rules can be applied in any order, though some orders may be quicker than others (especially when the constraints cannot be made BC). Similar rules for set variables are given in [8].

It is easy to see that the application of these rules terminates either with domains that are at a fixed point or with failure. Indeed, these rules terminate either with the unique BC domains or, if the problem cannot be made BC, fail, in both cases independent of the order of application of the rules.

Theorem 3.6 *If a set of constraints in normal form can be made BC, these inference rules reach a unique fixed point in which domains are BC. If the constraints cannot be made BC, the inference rules terminate with failure. Both take at most $O(enm^2)$ time where e is the number of constraints, n is the number of variables, and m is the maximum cardinality of the multiset variables.*

Proof Each inference rule tightens the upper and lower bounds of a variable or flags failure. The rules must therefore reach a fixed point or fail.

Suppose that we reach some fixed points applying these rules to a set of constraints in normal form. The proof uses case analysis on the type of constraint. Consider, for example, a constraint of the form $X = Y \cup Z$ (3). We consider each of the multiset variables in turn and show that their domains are BC. For the variable X, as the inference rule tightening X's upper and lower bounds is at a fixed point, it must be the case that $glb(Y) \cup glb(Z) \subseteq glb(X)$, $lub(X) \subseteq lub(Y) \cup lub(Z)$, and $glb(X) \subseteq lub(X)$. The assignment $X = glb(X)$, $Y = lub(Y) \cap glb(X)$, and $Z = lub(Z) \cap glb(X)$ will satisfy the constraint $X = Y \cup Z$ and the conditions that $glb(Y) \subseteq Y \subseteq lub(Y)$ and $glb(Z) \subseteq Z \subseteq lub(Z)$. Similarly, the assignment $X = lub(X)$, $Y = lub(Y) \cap lub(X)$, and $Z = lub(Z) \cap lub(X)$ will satisfy the constraint $X = Y \cup Z$ and the conditions that $glb(Y) \subseteq Y \subseteq lub(Y)$ and $glb(Z) \subseteq Z \subseteq$

$lub(Z)$. Hence, X's domain is BC. Similar arguments hold for the domains of Y and Z, as well as for the other types of constraints. Hence, if the rules terminate at a fixed point, the resulting domains are BC.

We now prove that, if the domains in the problem can be made BC, the rules terminate at this fixed point. Consider a problem that can be made BC, and its unique BC domains. The proof again uses extensive case analysis on the type of constraint. Consider, for instance, the constraint $X = Y \cup Z$ and the BC domains for X, Y, and Z. To prove that the rules terminate at this fixed point, we assume that an inference rule can still narrow a domain or flag failure. There are five cases corresponding to the five different inference rules associated with this constraint. In the first, the inference rule narrows the least upper bound of Y by removing one or more values. Suppose one of these removed values is m. Let $occ(m, X)$, $occ(m, Y)$, and $occ(m, Z)$ be the number of occurrences of m in X, Y, and Z respectively. As m is pruned by this inference rule, $\max(occ(m, Y)) > \max(occ(m, X))$. The original multiset variables are not therefore BC (which is a contradiction). Hence, there can be no value m removed and this inference rule is at a fixed point if the domains are BC. Similar arguments hold for the inference rules (1), (2), (3), and (4).

These rules therefore terminate at a fixed point if and only if the resulting domains are BC. As the rules must terminate either at a fixed point or by flagging failure, it follows that the rules flag failure if and only if the problem cannot be made BC. As each rule tightens the bounds on a multiset, set, or finite domain occurrence variable, the worst case is when the rules tighten each bound by just one element at a time. We may therefore have to apply $O(nm)$ rules. To find which rule applies, we may have to go through each of the e constraints in turn. Associated with each type of constraints, a fixed number of rules can be tried. The cost of applying the inference rules is thus at most $O(enm)$ multiplied by the cost of applying a single inference rule. This last cost is dominated by the $O(m)$ cost to test (dis)equality or inclusion, and the $O(m)$ cost to perform one of the basic operations like union or difference. Hence, the total cost is $O(enm^2)$ in the worst case. □

4 Cardinality constraints and reasoning

Similar to sets, we define the cardinality of a multiset S, denoted as $|S|$, as the total number of elements in S. The cardinality $|S|$ can be modeled using the constraint $|S| = \sum_i occ(i, S)$ for all elements i in S. In this section, we discuss the benefits of incorporating cardinality information to a multiset variable and propose some refined inference rules.

4.1 Cardinality variable

In Section 3, we represent a multiset variable S as an occurrence vector $\langle occ(1, S), \ldots, occ(n, S)\rangle$ in which $occ(i, S)$ denotes the number of occurrences of element i in S. In fact, we can have another component, a *cardinality variable* C_S [1, 2], to denote the total number of elements in S, together with the occurrence vector $\langle occ(1, S), \ldots, occ(n, S)\rangle$. The domain of C_S is an interval bounded by its lower bound c_r and upper bound c_p (i.e., $D_{C_S} = [c_r, c_p]$).

Example 4.1 Suppose $n = 2$ and consider a multiset variable S whose components have the following domains: $D_{occ(1,S)} = [0, 2]$, $D_{occ(2,S)} = [0, 3]$ and $D_{C_S} = [0, 5]$. Then, we have (1) $s_r = \emptyset$, as the lower bounds of all the occurrence variables are 0; and (2) $s_p = \{\{1, 1, 2, 2, 2\}\}$, as the upper bounds of the occurrence variables of elements 1 and 2 are 2 and 3 respectively. The domain of S is in fact the multiset interval $[\emptyset, \{\{1, 1, 2, 2, 2\}\}]$ with cardinality bounded from 0 to 5.

Representing and reasoning about the cardinality of a multiset variable is beneficial. This can increase the expressiveness of models by posting constraints directly on the cardinality variables, and we can model the domain of a multiset variable in a more precise way. For example, instead of posting a reified constraint $|\sum_i occ(i, S)| < 5$ on the occurrence variables, we can post a constraint $C_S < 5$ on the cardinality variable. The propagation of reified constraints is usually weak in most constraint solvers.

To give a better representation of a multiset variable, we can combine the occurrence representation and the cardinality representation. The occurrence representation maintains the integer variables denoting the number of occurrences for each element in the multiset variable, while the cardinality representation maintains the integer variable denoting the total number of elements in the multiset variable. We shall use the abbreviations O and C for the occurrence and cardinality representation respectively. Thus, the combination of the occurrence and cardinality representations has the abbreviation O/C representation.

Theorem 4.1 *The occurrence/cardinality (O/C) representation is more expressive than the occurrence representation alone.*

Proof Consider the following sets of the set variable S_1: {1} and {2}. These can be represented exactly with an O/C representation (i.e., $D_{occ(1,S_1)} = [0, 1]$, $D_{occ(2,S_1)} = [0, 1]$, $D_{C_{S_1}} = [1, 1]$) but not with the occurrence representation alone as this would not limit the total number of elements S_1 can take and thus also include the sets $\emptyset$ and $\{1, 2\}$. Similarly, consider the multiset variable S_2 with multiset values $\{\{1, 1\}\}$, $\{\{1, 2\}\}$, and $\{\{2, 2\}\}$. These can be represented exactly with an O/C representation (i.e., $D_{occ(1,S_1)} = [0, 2]$, $D_{occ(2,S_1)} = [0, 2]$, $D_{C_{S_1}} = [2, 2]$) but not with an occurrence representation alone as this would also include the multisets $\emptyset$, $\{\{1\}\}$, $\{\{2\}\}$, $\{\{1, 1, 2\}\}$, $\{\{1, 2, 2\}\}$, and $\{\{1, 1, 2, 2\}\}$. Exact cardinality constraints are required if the variables are represented by the occurrence representation alone. □

4.2 Cardinality reasoning

When we incorporate cardinality information in multiset variables, we can relate not only the number of occurrences of each element, but also the cardinalities of the multisets. In the O/C representation, the number of occurrences of each element is closely related to the cardinality of a multiset variable. Thus, we also need to reason on the cardinality constraints when we enforce BC on multiset constraints. In the following, we update the inference rules stated in Section 3.4. For each rule, we will explain the changes in the constraint store involving cardinality variables, which are adopted from Azevedo and Barahona [1, 2]. Note that X, Y, and Z are multiset

variables or ground multisets. C_X, C_Y, and C_Z are the cardinality variables of X, Y, and Z respectively. N is an integer variable or an integer constant.

Equality constraint ($X = Y$)

When X and Y are forced to be equal, their cardinalities are also equal.

$$\overline{\{X = Y\} \mapsto \{occ(i, X) = occ(i, Y), C_X = C_Y\}} \tag{8}$$

Subset constraint ($X \subseteq Y$)

When Y contains X, C_Y is also greater than or equal to C_X.

$$\overline{\{X \subseteq Y\} \mapsto \{occ(i, X) \leq occ(i, Y), C_X \leq C_Y\}} \tag{9}$$

Union constraint ($X \cup Y = Z$)

When Z is the union of X and Y, C_Z is smaller than or equal to $C_X + C_Y$. On the other hand, the lower bound of C_Z can be obtained from the maximum of the following two cases: (1) Suppose Z contains X (i.e., $X \subseteq Z$), Z will have at least C_X elements. We can safely add the elements which appear in Y but not in X (i.e., $y_r \setminus x_p$) to Z because Z is the multiset union and it takes all elements in both X and Y. Thus, $C_Z \geq C_X + |y_r \setminus x_p|$. (2) Similarly, we can add the elements in $(x_r \setminus y_p)$ to Z if Z contains Y. Thus, $C_Z \geq C_Y + |x_r \setminus y_p|$.

$$\overline{\begin{aligned}\{X \cup Y = Z\} \mapsto \{&occ(i, Z) = \max(occ(i, X), occ(i, Y)),\\ &occ(i, X) \leq occ(i, Z), occ(i, Y) \leq occ(i, Z),\\ &C_Z \leq C_X + C_Y,\\ &C_Z \geq \max(C_X + |y_r \setminus x_p|, C_Y + |x_r \setminus y_p|)\}\end{aligned}} \tag{10}$$

Union-plus constraint ($X \uplus Y = Z$)

When Z is the union-plus of X and Y, C_Z equals $C_X + C_Y$ because union-plus sums up all the elements in both X and Y.

$$\overline{\begin{aligned}\{X \uplus Y = Z\} \mapsto \{&occ(i, Z) = occ(i, X) + occ(i, Y),\\ &occ(i, X) \leq occ(i, Z), occ(i, Y) \leq occ(i, Z),\\ &C_Z = C_X + C_Y\}\end{aligned}} \tag{11}$$

Intersection constraint ($X \cap Y = Z$)

If Z is the intersection of X and Y, then C_Z is smaller than or equal to both C_X and C_Y. The upper bound of C_Z can be obtained from the minimum of the following two cases: (1) For the elements existing only in x_r but not in y_p (i.e., $x_r \setminus y_p$), they must not be part of the intersection. We can safely subtract these elements from C_X,

resulting $C_Z \geq C_X - |x_r \setminus y_p|$. (2) Similarly, we can subtract the elements that exist in y_r but not in x_p (i.e., $y_r \setminus x_p$) from C_Y, resulting $C_Z \geq C_Y - |y_r \setminus x_p|$.

$$\frac{}{\begin{array}{l}\{X \cap Y = Z\} \mapsto \{occ(i, Z) = \min(occ(i, X), occ(i, Y)),\\ \qquad occ(i, Z) \leq occ(i, X), occ(i, Z) \leq occ(i, Y),\\ \qquad C_Z \leq \min(C_X, C_Y),\\ \qquad C_Z \geq \min(C_X - |x_r \setminus y_p|, C_Y - |y_r \setminus x_p|)\}\end{array}} \tag{12}$$

Cardinality constraint ($C_X = N$)

When the cardinality of X equals N, the cardinality variable C_X also equals N.

$$\frac{}{\{|X| = N\} \mapsto \{C_X = N\}} \tag{13}$$

Failure rules

A failure can be detected (1) when the lower bound x_r is not included in the upper bound x_p, or (2) when the domain of the cardinality variable D_{C_X} becomes empty.

$$(X \text{ changed bounds}) \quad \frac{\text{not}(x_r \subseteq x_p)}{\{\} \mapsto \text{fail}} \quad \frac{D_{C_X} = \emptyset}{\{\} \mapsto \text{fail}} \tag{14}$$

Similar to the inference rules stated in Section 3.4, each of the above rules also tightens an upper and/or lower bound on a variable. Thus, the rules can terminate either with domains at a fixed point or by flagging failure independent of the order of applying the rules.

Besides giving a more expressive representation of multiset variables and multiset constraints, cardinality reasoning can also increase the pruning affected by a bounds consistency propagator. Following Debruyne and Bessière, we define the notion of *stronger* between two local consistencies A and B. A is *stronger* [4] than B ($A \geq B$) if in any CSP in which A holds, then B holds too.

Theorem 4.2 *BC on a constraint containing multiset variables using the occurrence/cardinality (O/C) representation is strictly stronger than BC on that using the occurrence representation alone.*

Proof Clearly both are at least as strong. To show strictness, consider $X \cap Y = \emptyset$, $\emptyset \subseteq X$, $Y \subseteq \{\{1, 2, 2\}\}$, $|X| = |Y| = 2$. Enforcing BC on the intersection constraint in which X and Y are using the O/C representation demonstrates that the problem is unsatisfiable. BC on that using the occurrence representation alone, on the other hand, does not detect this. □

Example 4.2 Suppose we have three multiset variables S_1, S_2, and S_3, where $D_{S_1} = [\emptyset, \{\{1, 1, 2\}\}]$, $D_{S_2} = [\{\{1\}\}, \{\{1, 1, 2\}\}]$, and $D_{S_3} = [\{\{1\}\}, \{\{1, 1, 2\}\}]$. Each set variable is associated with an integer variable denoting their cardinality: $D_{C_{S_1}} = D_{|S_1|} = [0, 3]$, $D_{C_{S_2}} = D_{|S_2|} = [1, 3]$, and $D_{C_{S_3}} = D_{|S_3|} = [2, 2]$. In this problem, there is a union constraint relating the three set variables: $S_1 = S_2 \cup S_3$.

After enforcing set bounds consistency, the element 1 is added to $glb(S)$ of S_1 because both S_2 and S_3 must contain element 1. The bounds of the cardinality of S_1

is changed from [0, 3] to [1, 3]. However, it is known that S_3 has exactly two elements. It should be inferred that the cardinality of S_1 can never lower than two elements. Thus, we can further update the bounds of the cardinality of S_1 to [2, 3].

Example 4.2 shows how cardinality reasoning can further reduce the domain of the cardinality of a multiset variable. Such process can help to improve the efficiency of problem solving as the search space is reduced. Sometimes, a failure can be detected at an earlier state than simply using bounds reasoning. However, we have to be careful that cardinality reasoning introduces additional complexity. In particular, enforcing bounds consistency can go from being polynomial to being NP-hard.

Theorem 4.3 *Given a set or multiset variable S represented by bounds, there exists a unary constraint $C(S)$ such that enforcing BC on $C(S)$ is polynomial but enforcing BC on $C(S) \wedge |S| = c$ is NP-hard where c is a constant.*

Proof For a set variable S, we suppose the set contains elements which represent either clauses or literals in these clauses. For instance, we might have S contain numbers and have an encoding scheme to represent each possible clause or literal by an unique number. Given any such S, we can compute m, the number of clauses in S and n, the number of variables that appear positively or negatively in clauses in S.

We define $C(S)$ to be satisfied iff $|S| \neq n + m$ or ($|S| = n + m$, S contains literals which satisfy all the clauses in S but does not contain two contradictory literals). Note that C is polynomial to check. Enforcing BC on $C(S)$ is also polynomial. There are three cases. In the first case, $|glb(S)| = |lub(S)| = n + m$. Enforcing BC simply needs to check if the literals in S satisfy the clauses in S. In the second case, $|glb(S)| = n + m - 1$ and $|lub(S)| = n + m$. Enforcing BC again simply requires us to check if the literals in $lub(S)$ satisfy the clauses in $lub(S)$. If so, S is already BC. Otherwise, we reduce $lub(S)$ to equal $glb(S)$ and the bounds now have support. In the third case, which covers all other situations, S is already BC.

With an additional cardinality bound, $|S| = c$, we show that enforcing BC is NP-hard by a reduction from 3-SAT. We set the $lub(S)$ to be the clauses of the 3-SAT problem that we wish to decide, and $glb(S)$ to be $lub(S)$ plus every possible literal in the 3-SAT clauses. We also set $c = n + m$. Then finding a bound support is equivalent to deciding if the 3-SAT problem is satisfiable.

For a multiset variable S, we repeat the construction of C as above but treat any multiset as the set of distinct elements it contains. □

5 Variety constraints and reasoning

The cardinality $|S|$ of a multiset variable S reveals only the total number of elements, but not the number of *distinct* elements, which we define as *variety* and is denoted by $\|S\|$. Note that $\|S\| \leq |S|$. In fact, S degenerates to a set when $|S| = \|S\|$ (i.e., the number of occurrences of all elements equals one). We can model the variety V_S using the constraints $V_S = \sum_i (occ(i, S) > 0)$ for each distinct element i in S. The propagation of such reified constraint is usually weak in most constraint solvers.

In this section, we discuss the benefits of incorporating variety information to a multiset variable. Then, we formally define a multiset variable with the additional variety information. Subsequently, we propose some inference rules which further improve over those incorporated with cardinality information. We also demonstrate how variety can help increase propagation for multiset constraints using variety reasoning.

5.1 Variety variable

On top of the occurrence vector [22] and the cardinality variable [1, 2], we further incorporate the variety information by adding the third component, a *variety variable* V_S to model the number of *distinct* elements in S (denoted $\|S\|$). Similar to C_S, the domain of V_S is also denoted as the interval bounded by its lower bound v_r and upper bound v_p (i.e., $D_{V_S} = [v_r, v_p]$). In the rest of the paper, we shall use the abbreviations V for the variety representation, which maintains an integer variable representing the number of elements of the multiset when viewed as a set. The combination of the occurrence, cardinality, and variety representations has the abbreviation O/C/V representation. The O/C/V representation is also called the *hybrid representation*.

Example 5.1 Suppose $n = 4$ and consider a multiset variable S whose components have the following domains: $D_{occ(1,S)} = [0, 1]$, $D_{occ(2,S)} = [0, 2]$, $D_{occ(3,S)} = [0, 3]$, $D_{occ(4,S)} = [0, 1]$, $D_{C_S} = [0, 7]$, and $D_{V_S} = [0, 4]$. Then, we have (1) $s_r = \emptyset$, as the lower bounds of all the occurrence variables are 0; and (2) $s_p = \{\{1, 2, 2, 3, 3, 3, 4\}\}$, as the upper bounds of the occurrence variables of elements 1, 2, 3, and 4 are 1, 2, 3, and 1 respectively. The domain of S is in fact the multiset interval $[\emptyset, \{\{1, 2, 2, 3, 3, 3, 4\}\}]$ with C_S bounded from 0 to 7, and V_S bounded from 0 to 4.

Similar to the cardinality variable, introducing a variety variable to the representation allows us to model the domain of a multiset variable in a more precise way (although still inexact). Consider S in the previous example and suppose we are interested in only the domain values whose varieties are 1. Without the variety variable V_S, we can only set D_{C_S} to $[1, 3]$. This domain accepts, for example, the multiset $\{\{1, 2\}\}$, which obviously should not be included. However, with V_S, we can simply set $D_{V_S} = [1, 1]$ to further remove the multisets which contain more than one kind of elements. This essentially models $D_S = \{\{\{1\}\}, \{\{2\}\}, \{\{2, 2\}\}, \{\{3\}\}, \{\{3, 3\}\}, \{\{3, 3, 3\}\}, \{\{4\}\}\}$, a much more precise representation.

Another advantage of introducing variety variables is that we can increase the expressiveness of models by posting constraints directly on them. For example, in the template design problem, we may want to restrict a template T to have at most three distinct designs in its slots. Using our representation, posting a variety constraint $\|T\| \leq 3$ simply means to propagate the simple unary constraint $V_T \leq 3$. Without V_T, we need to post a number of reified constraints to model the requirement, which may hinder propagation. The advantage becomes more obvious when the form of the variety constraints is more complicated, e.g., $\|T_1\| + \|T_2\| \leq 4$.

Example 5.2 Consider two multiset variables S_1 and S_2 where $D_{S_1} = [\{\{1\}\}, \{\{1, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,S_1)} = [1, 1]$, $D_{occ(2,S_1)} = [0, 1]$, $D_{occ(3,S_1)} = [0, 2]$), $D_{C_{S_1}} = [3, 3]$, $D_{V_{S_1}} = [1, 3]$, and $D_{S_2} = [\{\{1\}\}, \{\{1, 4, 5, 5\}\}]$ (i.e., $D_{occ(1,S_2)} = [1, 1]$, $D_{occ(4,S_2)} = [0, 1]$,

$D_{occ(5,S_2)} = [0, 2])$, $D_{C_{S_2}} = [3, 3]$, $D_{V_{S_2}} = [1, 3]$. Suppose we now post a constraint in which the variety of the union-plus of S_1 and S_2 is not greater than 3 (i.e., $\|S_1 \uplus S_2\| \leq 3$). Reasoning on this constraint reveals that S_1 and S_2 cannot contain elements 2 and 4 respectively, because the cardinalities of both S_1 and S_2 must be 3 and their union-plus can contain at most three different elements. S_1 and S_2 *should* then be bound to $\{\{1, 3, 3\}\}$ and $\{\{1, 5, 5\}\}$ respectively. However, using the reified constraint $\sum_{i=1}^{5}(occ(i, S_1) + occ(i, S_2) > 0) \leq 3$, the domains of S_1 and S_2 remain unchanged.

Theorem 5.1 *The occurrence/cardinality/variety (O/C/V) representation is more expressive than the occurrence/cardinality (O/C) representation. Similarly, the occurrence/variety (O/V) representation is more expressive than the occurrence representation.*

Proof Consider the following sets: {1}, {2}. These can be represented exactly with an O/V representation but not with the occurrence representation as this would also include the sets ∅ and {1, 2}. Similarly, consider the multisets $\{\{1, 1, 2\}\}$, $\{\{1, 1, 3\}\}$, $\{\{1, 2, 2\}\}$, $\{\{1, 3, 3\}\}$, $\{\{2, 2, 3\}\}$, and $\{\{2, 3, 3\}\}$ in which the elements 1, 2, and 3 can appear at most twice. These can be represented exactly with an O/C/V representation but not with an O/C representation as this would also include the multiset $\{\{1, 2, 3\}\}$. □

By exploiting the relationships between the three components of a multiset variable, we propose a number of inference rules to strengthen propagation. In the next subsection, we shall systematically enumerate the possible relationships.

5.2 Inferences within one multiset variable

Upon creation of a multiset variable S, the vector of occurrence variables $\langle occ(1, S), \ldots, occ(n, S)\rangle$, the cardinality variable C_S, and the variety variable V_S will also be created. A number of inference rules are subsequently maintained. Inferences occur between any two kinds of variables (i.e., between $occ(i, S)$ and C_S, between $occ(i, S)$ and V_S, or between C_S and V_S), or among all three of them.

5.2.1 Inferences between occ(i, S) and C_S

The cardinality C_S must always remain inside the limits given by the multiset bounds s_r and s_p [1, 2]. (Recall that s_r and s_p can be computed using the occurrence variables.)

$$(S \text{ changed bounds}) \quad \frac{}{\{\} \mapsto \{C_S \geq |s_r|, C_S \leq |s_p|\}} \tag{15}$$

The cardinality C_S is the sum of the number of occurrences of all elements in a multiset variable. Thus, the number of occurrences of each element is updated when there are changes in the lower bound c_r and the upper bound c_p of the cardinality variable. Here, we only concern with the elements that have not yet been included in

the lower bound (i.e., $s_p \setminus s_r$, which is defined as $occ(i, s_p \setminus s_r) = \max\{0, occ(i, s_p) - occ(i, s_r)\}$ for all elements i).

$$(C_S \text{ changed bounds}) \quad \frac{}{\{\} \mapsto \{|K| - occ(i, K) \geq c_r - |s_r|, occ(i, K) \leq c_p - |s_r|\}} \quad (16)$$

where $K = s_p \setminus s_r$.

Example 5.3 Consider a multiset variable S where $D_S = [\{\{1, 1\}\}, \{\{1, 1, 1, 2, 2, 3\}\}]$ (i.e., $D_{occ(1,S)} = [2, 3]$, $D_{occ(2,S)} = [0, 2]$, $D_{occ(3,S)} = [0, 1]$), $D_{V_S} = [1, 3]$, and c_p is updated from 6 to 3 (i.e., $D_{C_S} = [2, 3]$). Since S can now have at most three elements of at most two different kinds, and there are already two 1s in $glb(S)$, only one more element (1, 2, or 3) may exist in S. Based on the inference rule, $K = s_p \setminus s_r = \{\{1, 1, 1, 2, 2, 3\}\} \setminus \{\{1, 1\}\} = \{\{1, 2, 2, 3\}\}$ and $occ(i, K) \leq \beta - |s_r| = 3 - 2 = 1$ for $i = 1, 2, 3$. Thus, one of the 2s is removed, resulting $D_{occ(2,S)} = [0, 1]$ and $D_S = [\{\{1, 1\}\}, \{\{1, 1, 1, 2, 3\}\}]$.

5.2.2 Inferences between $occ(i, S)$ and V_S

The variety V_S must always remain inside the limits given by the multiset bounds. This is generalized from the inferences between $occ(i, S)$ and C_S [1, 2].

$$(S \text{ changed bounds}) \quad \frac{}{\{\} \mapsto \{V_S \geq \|s_r\|, V_S \leq \|s_p\|\}} \quad (17)$$

The occurrence of each element $occ(i, S)$ will be updated only when V_S is bound and equals either $\|s_r\|$ or $\|s_p\|$. When the variety V_S is fixed and equals $\|s_r\|$, any element i that is not in s_r (i.e., $occ_r(i, S) = 0$) has to be removed (i.e., $occ(i, S) = 0$ for those i). On the other hand, if $V_S = \|s_p\|$, then each element in s_p (i.e., $occ_p(i, S) > 0$) must occur at least once in S (i.e., $occ(i, S) > 0$ for those i).

$$(V_S \text{ is bound}) \quad \frac{V_S = \|s_r\|, occ_r(i, S) = 0}{\{\} \mapsto \{occ(i, S) = 0\}} \quad \frac{V_S = \|s_p\|, occ_p(i, S) > 0}{\{\} \mapsto \{occ(i, S) > 0\}} \quad (18)$$

Example 5.4 Consider a multiset variable S where $D_S = [\{\{1, 1\}\}, \{\{1, 1, 2, 2, 3\}\}]$ (i.e., $D_{occ(1,S)} = [2, 2]$, $D_{occ(2,S)} = [0, 2]$, $D_{occ(3,S)} = [0, 1]$), $D_{C_S} = [2, 5]$, and V_S is bound to 1. Since $V_S = \|s_r\|$ (i.e., $1 = \|\{\{1, 1\}\}\|$) and the elements 2 and 3 are not yet in s_r, they will not exist in S, resulting $occ(2, S) = occ(3, S) = 0$.

Consider the same multiset variable S but V_S is now bound to 3. Since $V_S = \|s_p\|$ (i.e., $3 = \|\{\{1, 1, 2, 2, 3\}\}\|$) and the elements 2 and 3 are not yet in s_r, at least one occurrence of 2 and 3 has to be added to their lower bound, resulting $occ_r(2, S) = occ_r(3, S) = 1$. Here, $occ_r(1, S)$ remains unchanged because the element 1 is already in its lower bound.

5.2.3 Inferences between C_S and V_S

The variety of a multiset must always be smaller than or equal to its cardinality at both limits because cardinality counts repeated elements but variety does not.

$$(C_S \text{ changed upper bound}) \quad \frac{}{\{\} \mapsto \{V_S \leq c_p\}} \quad (19)$$

$$(V_S \text{ changed lower bound}) \quad \frac{}{\{\} \mapsto \{C_S \geq v_r\}} \tag{20}$$

5.2.4 Inferences among occ(i, S), C_S, and V_S

When any two of occurrences $occ(i, S)$, cardinality C_S, and variety V_S change their bounds, the remaining one has to be updated as well. This kind of inferences leads to stronger constraint propagation than those between the pairwise ones (i.e., between $occ(i, S)$ and C_S, between $occ(i, S)$ and V_S, and between C_S and V_S).

When the occurrences $occ(i, S)$ and the variety V_S change their bounds, the cardinality C_S will be adjusted accordingly to fulfill the requirements on V_S based on the elements existing in $s_p \setminus s_r$.

$$(S \text{ or } V_S \text{ changed bounds}) \quad \frac{}{\{\} \mapsto \{C_S \geq |s_r| + (v_r - \|s_r\|), C_S \leq |s_r| + a\}} \tag{21}$$
$$\text{where } a = \max(|b| : b \subseteq (s_p \setminus s_r) \wedge \|b \uplus s_r\| = v_p).$$

Example 5.5 Consider a multiset variable S which updates its bounds to $D_S = [\{\{1, 1\}\}, \{\{1, 1, 1, 2, 2, 3\}\}]$ (i.e., $D_{occ(1,S)} = [2, 3]$, $D_{occ(2,S)} = [0, 2]$, $D_{occ(3,S)} = [0, 1]$), $D_{C_S} = [2, 6]$, and $D_{V_S} = [2, 3]$. Since S must contain at least two different kinds of elements, besides element 1, either element 2 or 3 has to be included in S. This leads to an increase in c_r although the exact addition has not yet taken place. Based on the inference rule, $C_S \geq |s_r| + (v_r - \|s_r\|) = 2 + 2 - 1 = 3$. Thus, D_{C_S} is updated to $[3, 6]$.

To find a, the subset $s_p \setminus s_r$, which fulfills the condition $\|b \uplus s_r\| = \beta$, is first extracted. The possible elements are then ordered. Thus, the complexity is bounded by the sorting procedure $O(n \log n)$, where n is the number of distinct elements in S.

Similarly, when the occurrences $occ(i, S)$ and the cardinality C_S change their bounds, the variety V_S will be adjusted accordingly to fulfill the requirements on C_S based on the elements existing in $s_p \setminus s_r$.

$$\begin{aligned}&(S \text{ or } C_S \text{ changed bounds}) \quad \frac{}{\{\} \mapsto \{V_S \geq a, V_S \leq \|s_r\| + c\}} \\ &\text{where } a = \min(\|s_r \uplus b\| : b \subseteq (s_p \setminus s_r) \wedge |b \uplus s_r| = c_r), \text{ and} \\ &\qquad c = \max(\|d\| : d \subseteq (s_p \setminus s_r) \wedge \|d \uplus s_r\| > \|s_r\| \wedge |d \uplus s_r| = c_p).\end{aligned} \tag{22}$$

The values of a and c can be obtained using the same way as finding a in the previous inference rule, but with different conditions. Thus, the complexity for this inference rule as a whole is also bounded by the sorting procedure $O(n \log n)$, where n is the number of distinct elements in S.

Example 5.6 Consider a multiset variable S where $D_S = [\{\{1, 1\}\}, \{\{1, 1, 1, 2, 2, 3\}\}]$ (i.e., $occ(1, S) = [2, 3]$, $occ(2, S) = [0, 2]$, $occ(3, S) = [0, 1]$), $D_{C_S} = [4, 6]$, and $D_{V_S} = [1, 3]$. Here, S must contain at least four elements (i.e., $c_r = 4$). With the current v_r, S can only have at most three 1s and one more element is needed to reach c_r. Other elements, which lead to a minimal change in v_r, are selected from their upper bounds. Based on the inference rule, b refers to a subset of $(s_p \setminus s_r)$ where $D_{(s_p \setminus s_r)} = [\emptyset, \{\{1, 2, 2, 3\}\}]$ and the cardinality of $b \uplus s_r$ equals c_r (i.e., 4). Thus, v_r equals the minimum variety of the union of s_r and a possible b (i.e., $\|\{\{1, 1\}\} \cup \{\{1, 2\}\}\| = 2$). V_S is updated to $[2, 3]$.

When the cardinality C_S is fixed and equals either $|s_r|$ or $|s_p|$, S can be set to the corresponding bound (i.e., all occurrences $occ(i, S)$ can be fixed) and V_S can also be bound accordingly.

$$(C_S \text{ is bound}) \quad \frac{C_S = |s_r|}{\{\} \mapsto \{S = s_r, V_S = \|s_r\|\}} \quad \frac{C_S = |s_p|}{\{\} \mapsto \{S = s_p, V_S = \|s_p\|\}} \tag{23}$$

When both the cardinality C_S and the variety V_S are bound to the same value α, S degenerates to a set. Thus, the occurrence of each element $occ(i, S)$ will be at most one.

$$(C_S \text{ and } V_S \text{ are bound and equal}) \quad \frac{C_S = V_S = \alpha}{\{\} \mapsto \{occ(i, S) \leq 1\}} \tag{24}$$

Failure

A failure can be detected when any one of the conditions is true: (1) the lower bound s_r is not included in the upper bound s_p; or (2) the domain of the cardinality variable D_{C_S} becomes empty; or (3) the domain of the variety variable D_{V_S} becomes empty.

$$(S, C_S, \text{or } V_S \text{ changed bounds}) \quad \frac{\text{not}(s_r \subseteq s_p)}{\{\} \mapsto \text{fail}} \quad \frac{D_{C_S} = \emptyset}{\{\} \mapsto \text{fail}} \quad \frac{D_{V_S} = \emptyset}{\{\} \mapsto \text{fail}} \tag{25}$$

5.2.5 Discussions

The inference rules described here are incomplete in the sense that they do not prune all possible values from the occurrence, cardinality, or variety variables.

Consider the cardinality constraint $card(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S)$ which ensures that C_S is the cardinality of the multiset S represented by the occurrence vector $\langle occ(1, S), \ldots, occ(m, S)\rangle$ (i.e., $C_S = \sum_{i=1}^{m} occ(i, S)$). Enforcing BC on such a constraint is polynomial since enforcing BC on such a sum is polynomial. However, enforcing GAC on such a constraint is NP-hard.

Theorem 5.2 *Enforcing GAC on* $card(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S)$ *is NP-hard.*

Proof We reduce subset sum to finding a support for $card(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S)$. Suppose we wish to find a subset of $S = \{a_i | 1 \leq i \leq m\}$ with sum s. Let $occ(i, S) = \{0, a_i\}$ and $C_S = s$. There exists a support for $card(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S)$ iff there exists a subset of S with sum s. □

On the other hand, consider the variety constraint, $variety(\langle occ(1, S), \ldots, occ(m, S)\rangle, V_S)$ which ensures that V_S is the variety of the multiset S represented by the occurrence vector $\langle occ(1, S), \ldots, occ(m, S)\rangle$ (i.e., $V_S = \sum_{i=1}^{m} (occ(i, S) > 0)$). Enforcing GAC on such a constraint is polynomial. Thus, we see that variety reasoning can be easy but cardinality reasoning is hard.

Theorem 5.3 *Enforcing GAC on* $variety(\langle occ(1, S), \ldots, occ(m, S)\rangle, V_S)$ *is polynomial.*

Proof We prove that GAC on $variety(\langle occ(1, S), \ldots, occ(m, S)\rangle, V_S)$ is equivalent to BC on the decomposition into $(occ(i, S) > 0) \leftrightarrow B_i$, and $V_S = \sum_{i=1}^{m} B_i$. The

constraint graph of this decomposition is acyclic. Hence GAC on the variety constraint is equivalent to GAC on the decomposition. GAC on $(occ(i, S) > 0) \leftrightarrow B_i$ is equivalent to BC on $(occ(i, S) > 0) \leftrightarrow B_i$ as this constraint only prunes at the lower bound of $occ(i, S)$. Finally, we prove GAC on $V_S = \sum_{i=1}^{m} B_i$ is equivalent to BC on $V_S = \sum_{i=1}^{m} B_i$. Clearly if $V_S = \sum_{i=1}^{m} B_i$ is GAC then it must be BC. For the reverse, suppose $V_S = \sum_{i=1}^{m} B_i$ is BC. Consider any value s in the domain of V_S between its upper and lower bounds. We can construct a support by assigning 1 to the $lb(V_S)$ variables with $dom(B_i) = \{1\}$ as well as to the $s - lb(V_S)$ variables with $dom(B_i) = \{0, 1\}$, and assigning 0 to the other $m - s$ variables. Thus, every value between upper and lower bounds of V_S has support. Hence, if $V_S = \sum_{i=1}^{m} B_i$ is BC then it is GAC. Note that BC is polynomial to enforce on each constraint in the decomposition. □

In fact, we can combine the cardinality and variety constraints to form an intra-variable constraint $intra(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S, V_S)$ which ensures that C_S and V_S are the cardinality and variety of the multiset S represented by the occurrence vector $\langle occ(1, S), \ldots, occ(m, S)\rangle$ respectively. Enforcing GAC on such a constraint is thus at least NP-hard.

Theorem 5.4 *Enforcing GAC on* $intra(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S, V_S)$ *is NP-hard.*

Proof The intra-variable constraint $intra(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S, V_S)$ can be decomposed to the cardinality constraint $card(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S)$ and the variety constraint $variety(\langle occ(1, S), \ldots, occ(m, S)\rangle, V_S)$. Thus, by Theorems 5.2 and 5.3, enforcing GAC on $intra(\langle occ(1, S), \ldots, occ(m, S)\rangle, C_S, V_S)$ is at least NP-hard. □

Our primary aim is not for completeness, but for inference rules that are efficiently implementable. Nonetheless, the inference rules as a whole maintain more than BC.

Theorem 5.5 *The inference rules* (1) *between* $occ(i, S)$ *and* C_S, (2) *between* $occ(i, S)$ *and* V_S, (3) *between* C_S *and* V_S, (4) *among* $occ(i, S)$, C_S, *and* V_S, *and* (5) *for failure collectively enforce a consistency level strictly stronger than BC on the O/C/V representation.*

Proof Obviously, the inference rules (14)–(24) collectively enforce a consistency level at least as strong as BC. For strictness, Examples 5.5 and 5.6 under Section 5.2.4 show that given a domain which is already multiset bounds consistent, the inference rules can further tighten the bounds of C_S or V_S. Hence the result holds. □

5.3 Cardinality-variety reasoning

The previous subsection describes the inferences within one multiset variable. Besides giving a more expressive representation, variety reasoning can also increase the pruning achieved by a bound consistency propagator.

Theorem 5.6 *BC on a constraint containing multiset variables using the O/C/V representation is strictly stronger than BC on that using the O/C representation. Similarly, BC on a constraint containing multiset variables using the O/V representation is strictly stronger than BC on that using the occurrence representation.*

Proof Clearly both are at least as strong. To show strictness, consider $X \cap Y = \emptyset$, $\emptyset \subseteq X$, $Y \subseteq \{\{1, 1, 2, 2, 3, 3\}\}$, $|X| = |Y| = 2$, $\|X\| = \|Y\| = 2$. Enforcing BC on the intersection constraint in which X and Y are using the O/C/V representation demonstrates that the problem is unsatisfiable. BC on that using the O/C representation, on the other hand, does not detect this. Similarly, enforcing BC on the intersection constraint in which X and Y are using the O/V representation demonstrates that the problem is unsatisfiable, but BC on that using the occurrence representation does not detect this. □

In this subsection, we focus on propagation that occurs across different multiset variables. We give some constraint propagation rules that enforce bounds consistency on some common multiset constraints. Performing inferences on both cardinality and variety variables are known as *cardinality-variety reasoning*, which is a combination of cardinality reasoning and variety reasoning. For each multiset constraint, we use an example to show how the propagation rules are useful in increasing constraint propagation. In the rules, the changes in the constraint store involving variety variables are more generalized than those involving cardinality variables, which are adopted from Azevedo and Barahona [1, 2]. We will only explain the updates from the rules stated in Section 4.2. Again, the rules can terminate either with domains at a fixed point or by flagging failure independent of the order of applying them.

5.3.1 Equality constraint ($X = Y$)

If X and Y are forced to be equal, then their varieties are equal.

$$\frac{}{\{X = Y\} \mapsto \{occ(i, X) = occ(i, Y), C_X = C_Y, V_X = V_Y\}} \tag{26}$$

Example 5.7 Consider an equality constraint $X = Y$, where $n = 3$, $D_{occ(1,X)} = [0, 2]$, $D_{occ(2,X)} = [0, 2]$, $D_{occ(3,X)} = [0, 2]$, $D_{C_X} = [4, 4]$, $D_{V_X} = [2, 2]$, $D_{occ(1,Y)} = [0, 2]$, $D_{occ(2,Y)} = [0, 2]$, $D_{occ(3,Y)} = [0, 2]$, $D_{C_Y} = [4, 4]$, and $D_{V_Y} = [3, 3]$. Without the variety variables V_X and V_Y (and thus without variety reasoning), there are no prunings available. However, with variety reasoning, the problem fails immediately because when $X = Y$ (i.e., $occ(i, X) = occ(i, Y)$ for all elements i), $V_X = V_Y$ is obviously violated.

5.3.2 Subset constraint ($X \subseteq Y$)

If Y contains X, then V_Y is greater than or equal to V_X.

$$\frac{}{\{X \subseteq Y\} \mapsto \{occ(i, X) \leq occ(i, Y), C_X \leq C_Y, V_X \leq V_Y\}} \tag{27}$$

Example 5.8 Consider a subset constraint $X \subseteq Y$, where $D_X = [\emptyset, \{\{1, 1, 2, 2, 3, 3, 3\}\}]$ with cardinality 5 and variety 3 (i.e., $D_{occ(1,X)} = [0, 2]$, $D_{occ(2,X)} = [0, 2]$, $D_{occ(3,X)} = [0, 3]$), $D_{C_X} = [5, 5]$, $D_{V_X} = [3, 3]$, and $D_Y = [\emptyset, \{\{1, 1, 2, 2, 3, 3, 3\}\}]$ with cardinality 5 and variety 2 (i.e., $D_{occ(1,Y)} = [0, 2]$, $D_{occ(2,Y)} = [0, 2]$, $D_{occ(3,Y)} = [0, 3]$), $D_{C_Y} = [5, 5]$, $D_{V_Y} = [2, 2]$. With variety reasoning, the problem fails immediately because V_X can never be smaller than or equal to V_Y (i.e., $3 \not\leq 2$). Again, without variety reasoning, there are no available prunings.

5.3.3 Union constraint ($X \cup Y = Z$)

When Z is the union of X and Y, V_Z is smaller than or equal to $V_X + V_Y$. Similarly, the lower bound of V_Z can be obtained from the maximum of the following two cases: (1) $V_Z \geq V_X + \|y_r \setminus x_p\|$ and (2) $V_Z \geq V_Y + \|x_r \setminus y_p\|$.

$$\begin{aligned} \{X \cup Y = Z\} \mapsto \{&occ(i, Z) = \max(occ(i, X), occ(i, Y)), \\ &occ(i, X) \leq occ(i, Z), occ(i, Y) \leq occ(i, Z), \\ &C_Z \leq C_X + C_Y, V_Z \leq V_X + V_Y, \\ &C_Z \geq \max(C_X + |y_r \setminus x_p|, C_Y + |x_r \setminus y_p|), \\ &V_Z \geq \max(V_X + \|y_r \setminus x_p\|, V_Y + \|x_r \setminus y_p\|)\} \end{aligned} \tag{28}$$

Example 5.9 Consider a union constraint $X \cup Y = Z$, where $D_X = [\emptyset, \{\{1, 1, 2, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,X)} = [0, 2]$, $D_{occ(2,X)} = [0, 2]$, $D_{occ(3,X)} = [0, 2]$), $D_{C_X} = [1, 2]$, $D_{V_X} = [1, 1]$, $D_Y = [\emptyset, \{\{1, 1, 2, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,Y)} = [0, 2]$, $D_{occ(2,Y)} = [0, 2]$, $D_{occ(3,Y)} = [0, 2]$), $D_{C_Y} = [1, 2]$, $D_{V_Y} = [1, 1]$, and $D_Z = [\{\{1, 2, 3\}\}, \{\{1, 1, 2, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,Z)} = [1, 2]$, $D_{occ(2,Z)} = [1, 2]$, $D_{occ(3,Z)} = [1, 2]$), $D_{C_Z} = [3, 6]$, $D_{V_Z} = [3, 3]$. With variety reasoning, the problem fails immediately because V_Z can never be smaller than or equal to the sum of V_X and V_Y. Without reasoning on the three variety variables, the problem will not fail even when $3 \not\leq 1 + 1$.

5.3.4 Union-plus constraint ($X \uplus Y = Z$)

When Z is the union-plus of X and Y, V_Z is smaller than or equal to $V_X + V_Y$ because X and Y can contain the same kind of elements (i.e., $\|X\| + \|Y\| \neq \|X \uplus Y\|$). For the lower bound of V_Z, it can be obtained in the same way as in the union constraint.

$$\begin{aligned} \{X \uplus Y = Z\} \mapsto \{&occ(i, Z) = occ(i, X) + occ(i, Y), \\ &occ(i, X) \leq occ(i, Z), occ(i, Y) \leq occ(i, Z), \\ &C_Z = C_X + C_Y, V_Z \leq V_X + V_Y, \\ &V_Z \geq \max(V_X + \|y_r \setminus x_p\|, V_Y + \|x_r \setminus y_p\|)\} \end{aligned} \tag{29}$$

Example 5.10 Consider a union-plus constraint $X \uplus Y = Z$, where $D_X = [\emptyset, \{\{1, 1, 2, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,X)} = [0, 2]$, $D_{occ(2,X)} = [0, 2]$, $D_{occ(3,X)} = [0, 2]$), $D_{C_X} = [1, 2]$, $D_{V_X} = [1, 1]$, $D_Y = [\emptyset, \{\{1, 1, 2, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,Y)} = [0, 2]$, $D_{occ(2,Y)} = [0, 2]$, $D_{occ(3,Y)} = [0, 2]$), $D_{C_Y} = [1, 2]$, $D_{V_Y} = [1, 1]$, and $D_Z = [\{\{1, 2, 3\}\}, \{\{1, 1, 2, 2, 3, 3\}\}]$ (i.e., $D_{occ(1,Z)} = [1, 2]$, $D_{occ(2,Z)} = [1, 2]$, $D_{occ(3,Z)} = [1, 2]$), $D_{C_Z} = [3, 6]$, $D_{V_Z} = [3, 3]$. Variety reasoning fails the problem immediately because V_Z can never be

smaller than or equal to the sum of V_X and V_Y. Without reasoning on the three variety variables, the problem will not fail even when $3 \not\leq 1 + 1$.

5.3.5 Intersection constraint ($X \cap Y = Z$)

If Z is the intersection of X and Y, then V_Z is smaller than or equal to both V_X and V_Y. The upper bound of V_Z can be obtained from the minimum of the following two cases: (1) $V_Z \geq V_X - \|x_r \setminus y_p\|$ and (2) $V_Z \geq V_Y - \|y_r \setminus x_p\|$.

$$\begin{array}{l} \{X \cap Y = Z\} \mapsto \{occ(i, Z) = \min(occ(i, X), occ(i, Y)), \\ \qquad\qquad occ(i, Z) \leq occ(i, X), occ(i, Z) \leq occ(i, Y), \\ \qquad\qquad C_Z \leq \min(C_X, C_Y), V_Z \leq \min(V_X, V_Y), \\ \qquad\qquad C_Z \geq \min(C_X - |x_r \setminus y_p|, C_Y - |y_r \setminus x_p|), \\ \qquad\qquad V_Z \geq \min(V_X - \|x_r \setminus y_p\|, V_Y - \|y_r \setminus x_p\|)\} \end{array} \tag{30}$$

Example 5.11 Consider an intersection constraint $X \cap Y = Z$, where $D_X = [\emptyset, \{\!\{1, 1, 2, 2, 3, 3, 3\}\!\}]$ (i.e., $D_{occ(1,X)} = [0, 2]$, $D_{occ(2,X)} = [0, 2]$, $D_{occ(3,X)} = [0, 3]$), $D_{C_X} = [1, 3]$, $D_{V_Z} = [1, 1]$, $D_Y = [\emptyset, \{\!\{1, 1, 2, 2, 3, 3, 3\}\!\}]$ (i.e., $D_{occ(1,X)} = [0, 2]$, $D_{occ(2,X)} = [0, 2]$, $D_{occ(3,X)} = [0, 3]$), $D_{C_Y} = [1, 3]$, $D_{V_Y} = [1, 1]$, and $D_Z = [\emptyset, \{\!\{1, 2, 3, 3, 3\}\!\}]$ (i.e., $D_{occ(1,X)} = [0, 1]$, $D_{occ(2,X)} = [0, 1]$, $D_{occ(3,X)} = [0, 3]$), $D_{C_Z} = [2, 4]$, $D_{V_Z} = [2, 2]$. With variety reasoning, the problem fails immediately because V_Z can never be smaller than or equal to both V_X and V_Y. The problem will not fail without variety reasoning even when $2 \not\leq 1$.

5.3.6 Failure rules

Each of the above rules tightens an upper and/or lower bound on a variable. Thus, the rules can terminate by flagging failure on any variables under any one of the conditions stated in Section 5.2.4.

5.3.7 Discussions

Cardinality reasoning introduces additional complexity. In particular, enforcing bounds consistency can go from being polynomial to being NP-hard. It follows that variety reasoning on a single multiset variable can also introduce additional complexity. As with a set variable, adding variety reasoning can make enforcing bounds consistency on a unary constraint go from being polynomial to being NP-hard.

Theorem 5.7 *Given a multiset variable S represented by bounds, there exists a unary constraint $C(S)$ such that enforcing BC on $C(S)$ is polynomial but enforcing BC on $C(S) \wedge \|S\| = d$ is NP-hard where d is a constant.*

Proof Similar to Theorem 4.3, the result holds. □

5.3.8 Decomposition of multiset constraints

In Section 3.3, we discuss decomposing multiset constraints into a flattened normal form in which constraints are at most ternary and only of the form. When the

constraint is complicated and contains a lot of variables, the solver may not be able to handle them. We can simplify the constraint by decomposing it into normal form.

In general, decomposition into such a normal form hinders constraint propagation, whether we use just the occurrence representation or we use the O/C/V representation (Theorem 3.4). Under the simple restriction that there are no repeated occurrences of variables and we are using an occurrence representation for the multisets, we proved that decomposition does not hinder BC propagation (Theorem 3.5). However, with cardinality and variety reasoning, decomposition of complex multiset constraints does hinder propagation even when variables are not repeated.

Theorem 5.8 *With cardinality reasoning, BC on a unary constraint on a single set variable can be strictly stronger than BC on the decomposition into normal form.*

Proof Consider the unary constraints: $X \subseteq \{1, 2, 3\}$, $|X| = 2$, $(X \cup \{3\}) \neq \{1, 2, 3\}$. With cardinality information, enforcing BC on these unary constraints will set $\{3\} \subseteq X$. Consider the decomposition into: $X \subseteq \{1, 2, 3\}, |X| = 2, Y = (X \cup \{3\}), 1 \leq |Y| \leq 3, Y \neq \{1, 2, 3\}$. Enforcing BC on these unary constraints will set $\{3\} \subseteq Y \subseteq \{1, 2, 3\}$. However, enforcing BC on the decomposition will not set $\{3\} \subseteq X$. □

It follows therefore that, with variety reasoning, BC on a *single* multiset variable can be strictly stronger than BC on the decomposition into the normal form.

Theorem 5.9 *With variety reasoning, BC on a unary constraint on a single multiset variable can be strictly stronger than BC on the decomposition into normal form.*

Proof Consider the unary constraints: $X \subseteq \{\!\{1, 1, 2, 2, 3\}\!\}$, $\|X\| = 3$, $X \uplus \{3\} \neq \{\!\{1, 1, 2, 2, 3\}\!\}$. With variety information, enforcing BC on these unary constraints will set $\{3\} \subseteq X$. Consider the decomposition into: $X \subseteq \{\!\{1, 1, 2, 2, 3\}\!\}, \|X\| = 3, Y = (X \uplus \{3\}), 1 \leq \|Y\| \leq 3, Y \neq \{\!\{1, 1, 2, 2, 3\}\!\}$. Enforcing BC on these unary constraints will set $\{3\} \subseteq Y \subseteq \{\!\{1, 1, 2, 2, 3\}\!\}$. However, enforcing BC on the decomposition will not set $\{3\} \subseteq X$. □

Since cardinality-variety reasoning is a combination of cardinality reasoning and variety reasoning, decomposition of complex multiset constraints also hinders propagation.

Theorem 5.10 *With cardinality-variety reasoning, BC on a unary constraint on a single multiset variable can be strictly stronger than BC on the decomposition into normal form.*

Proof By Theorems 5.8 and 5.9, the result follows. □

6 Experimental results

To verify the feasibility and efficiency of our proposal, we implemented our multiset variable representation, the inference rules, and the multiset constraints in ILOG Solver 6.0 [11]. We use four benchmarks, which are introduced in later sections,

for the experiments. The experiments are run on a Sun Blade 2500 (2 × 1.6GHz US-IIIi) workstation with 2GB memory. We report the number of fails (i.e., the number of backtracks occurred in solving a model) and CPU time in seconds to find and prove the optimal solution for each instance. Comparisons are made among bounds consistency on the occurrence representation (BC) [22], bounds consistency with cardinality reasoning (BC+CR) [1, 2], and bounds consistency with cardinality-variety reasoning (BC+CR+VR) [15]. We use the reified constraint $C_S = \sum_i (occ(i, S))$ mentioned in Section 4 to model the cardinality variables in BC, and use the reified constraint $V_S = \sum_i (occ(i, S) > 0)$ mentioned in Section 5 to model the variety variables in both BC and BC+CR. The reified constraints are enforced by the built-in propagation algorithms in ILOG Solver instead of the inference rules. In the tables, the best number of fails and CPU time among the results for each instance are highlighted in bold. A cell labeled with "–" denotes a timeout after 30 minutes.

6.1 Template design problem

In the template design problem $T(t, s, d, c)$ (prob002 in CSPLib), we are given d designs to t printing templates subject to some constraints. Each template has a fixed number of slots s for the designs and there should be at least c copies for each design. Each multiset variable represents a template and its domain values are the possible combinations of designs which allow repetitions. This is an optimization problem which minimizes the total number of pressings so that the amount of extra copies is reduced as much as possible. To future increase problem difficulty, we impose a restriction to constrain each multiset to have at least certain varieties v.

Table 3 shows the experimental results of the template design problem. From the experimental results, BC+CR+VR always achieves the fewest number of fails. There is a more than 90 % reduction in the number of fails when compared to BC alone. The amount of reduction increases with the tightness of the variety constraints. When the variety of each multiset is restricted to be at least 4, enforcing BC+CR+VR can also reduce the search space up to 40 % when compared to BC+CR. For runtimes, BC+CR+VR is also always the fastest. The amount of reduction also increases as the tightness of the variety constraints.

This problem does not contain multiset constraints like subset constraints and union constraints. It only has cardinality constraints, variety constraints, and arithmetic constraints to calculate the total number of copies of each design when the templates undergo certain number of pressings. Thus, the reduction in search space is mainly caused by the inference rules maintained within one multiset variable, as described in Section 5.2.

6.2 Extended steiner system

While the standard Steiner system is only set-based, the extended version is an important and practical multiset problem in the area of information retrieval [3, 12, 21]. Solving the extended Steiner system can provide solutions to the problem of a multiset batch code. The extended Steiner System $ES(t, k, u)$ is a collection of b blocks. Each block is a k-element multiset drawn from a u-element set whose elements can be drawn multiple times. For every two blocks in the collection, the cardinality of their intersection must be smaller than t. For example, one possible

Table 3 Results of the template design problem

(t, s, d, c)	v	BC		BC+CR		BC+CR+VR	
		Fails	Runtime	Fails	Runtime	Fails	Runtime
(3,5,5,5)	1	233649	3.14	11133	0.62	**11121**	**0.59**
	2	212818	2.87	10324	0.54	**10140**	**0.52**
	3	123130	1.63	6753	0.34	**6147**	**0.3**
	4	26432	0.33	2222	0.09	**1476**	**0.07**
	5	1632	0.02	433	0.02	**5**	**0.01**
(3,5,5,10)	1	2539032	33.83	85110	4.62	**85040**	**4.48**
	2	2319945	30.89	78759	4.11	**77609**	**3.91**
	3	1345093	17.63	50082	2.45	**46342**	**2.25**
	4	277339	3.44	16282	0.63	**10920**	**0.49**
	5	15671	0.17	3291	0.09	**10**	**0**
(3,5,5,15)	1	9102083	121.67	281529	15.45	**281336**	**14.96**
	2	8326588	111.16	260664	13.9	**257058**	**13.23**
	3	4829837	63.7	164955	8.16	**152885**	**7.47**
	4	997448	12.45	53655	2.11	**35599**	**1.59**
	5	54472	0.6	10844	0.29	**15**	**0.01**
(3,5,5,20)	1	22517539	301.77	667807	36.54	**667357**	**35.46**
	2	20607017	275.38	618852	33.14	**610658**	**31.57**
	3	11938093	157.42	393117	19.53	**364171**	**17.99**
	4	2442592	30.51	127007	5.05	**84431**	**3.79**
	5	130714	1.44	25599	0.67	**20**	**0.01**
(3,5,5,25)	1	43475937	573.45	1169486	66.21	**1168662**	**65.69**
	2	39835008	523.72	1089385	59.73	**1073891**	**58.48**
	3	23211047	301	706443	35.87	**652984**	**34**
	4	4810503	59.02	241153	9.7	**159238**	**7.51**
	5	256721	2.78	49717	1.31	**25**	**0**
(3,5,5,30)	1	77975941	1027.61	2173073	120.65	**2171643**	**120.14**
	2	71420872	938.78	2017818	109.4	**1990416**	**106.69**
	3	41441367	537.04	1285591	64.95	**1191642**	**61.36**
	4	8474261	104.29	424357	17	**280259**	**13.12**
	5	445367	4.8	85790	2.24	**30**	**0.01**
(3,5,5,35)	1	122846798	1622.59	3282341	184.83	**3280160**	**183.69**
	2	112579328	1483.12	3054543	167.83	**3012501**	**163.78**
	3	65493473	851.02	1967637	100.41	**1820206**	**94.45**
	4	13493394	165.85	661341	26.65	**435978**	**20.57**
	5	708706	7.66	135749	3.56	**35**	**0.01**
(3,5,5,40)	1	–	–	4965403	278.81	**4962121**	**276.46**
	2	–	–	4620373	253.02	**4557769**	**247.03**
	3	98966039	1286.08	2973352	150.69	**2751269**	**142.88**
	4	20306681	250.37	992771	40.1	**654489**	**30.83**
	5	1059773	11.49	202545	5.32	**40**	**0**

Table 3 (continued)

(t, s, d, c)	v	BC		BC+CR		BC+CR+VR	
		Fails	Runtime	Fails	Runtime	Fails	Runtime
(3,5,5,45)	1	–	–	6933558	391.15	**6929008**	**388.63**
	2	–	–	6454916	354.7	**6365856**	**346.37**
	3	–	–	4161884	212.37	**3852027**	**201.01**
	4	29002238	358.35	1404672	56.5	**924778**	**43.65**
	5	1510550	16.4	288045	7.6	**45**	**0**
(3,5,5,50)	1	–	–	9588716	540.07	**9582430**	**536.34**
	2	–	–	8926108	490.34	**8804592**	**479.25**
	3	–	–	5752173	292.84	**5325408**	**277.18**
	4	39986053	492.2	1934077	77.82	**1273470**	**60.13**
	5	2074142	22.59	395038	10.4	**50**	**0**

solution for $ES(2, 3, 3)$ in 3 blocks is $\{\{1, 1, 2\}, \{2, 2, 3\}, \{3, 3, 1\}\}$. The intersection of $\{1, 1, 2\}$ and $\{2, 2, 3\}$ is $\{2\}$; the intersection of $\{1, 1, 2\}$ and $\{3, 3, 1\}$ is $\{1\}$; the intersection of $\{2, 2, 3\}$ and $\{3, 3, 1\}$ is $\{3\}$. All of them have size smaller than $t = 2$. In our experiments, we adapt the extended Steiner System to an optimization problem which maximizes the sum of the varieties of the multisets in a solution. To further increase problem difficulty, we also constrain each multiset to have at least certain varieties v.

Tables 4 and 5 show the experimental results of the maximization and the variety of each multiset is at least 2 and 3 respectively. Among the three propagation approaches, BC+CR+VR always achieves the fewest number of fails. There is a

Table 4 Maximization results of the extended Steiner systems

(t, k, u)	b	BC		BC+CR		BC+CR+VR	
		Fails	Runtime	Fails	Runtime	Fails	Runtime
(2, 3, 4)	2	307	0.01	83	**0**	**20**	**0**
	3	2266	0.04	518	0.02	**135**	**0**
	4	9530	0.2	2232	0.1	**737**	**0.04**
(2, 3, 5)	2	800	**0.01**	222	**0.01**	**12**	0.02
	3	13812	0.23	3079	0.11	**156**	**0.01**
	4	166064	3.56	33679	1.58	**3539**	**0.21**
	5	1185644	31.37	244547	14.41	**39930**	**3**
	6	4744639	152.77	1095106	78.97	**244430**	**22.67**
(2, 4, 4)	2	867	0.01	204	**0**	**70**	**0**
	3	6246	0.1	1330	0.04	**581**	**0.03**
	4	20425	0.42	4980	0.21	**2757**	**0.18**
(2, 4, 5)	2	2800	0.04	638	0.03	**202**	**0**
	3	64458	1.12	12636	0.46	**4603**	**0.21**
	4	627704	14.13	124611	5.91	**57329**	**3.66**
	5	2800951	79.37	637199	38.06	**356785**	**29.27**

The variety of each multiset is at least 2

Table 5 Maximization results of the extended Steiner systems

(t, k, u)	b	BC		BC+CR		BC+CR+VR	
		Fails	Runtime	Fails	Runtime	Fails	Runtime
(2,3,6)	2	1751	0.02	480	0.01	**8**	**0**
	3	14895	0.28	2992	0.12	**13**	**0.01**
	4	57449	1.33	9548	0.49	**17**	**0**
(2,4,6)	2	6072	0.09	1219	0.04	**204**	**0.02**
	3	184841	3.55	33165	1.34	**11709**	**0.59**
	4	848172	20.62	132400	7.04	**56762**	**3.89**
(3,4,4)	2	446	0.02	160	0.02	**26**	**0**
	3	2549	0.04	793	0.02	**135**	**0.01**
	4	9615	0.18	2646	0.11	**634**	**0.04**
(3,4,5)	2	1475	0.02	537	**0.01**	**67**	**0.01**
	3	35582	0.58	10913	0.4	**1831**	**0.07**
	4	591668	12.07	160724	7.65	**29938**	**1.66**
	5	6565175	160.67	1630805	96.76	**312397**	**22.7**
	6	48187790	1370.32	11387223	812	**2108410**	**194.99**
	7	–	–	–	–	**9813128**	**1125.01**
(3,4,6)	2	4122	0.05	1537	0.05	**24**	**0.01**
	3	80815	1.4	27386	1.17	**63**	**0.01**
	4	3994239	87.12	1187476	66.63	**4134**	**0.25**
	5	–	–	–	–	**370386**	**26.52**
	6	–	–	–	–	**17854829**	**1562.45**

The variety of each multiset is at least 3

more than 90 % reduction in number of fails when compared to BC alone, and more than 50 % reduction when compared to BC+CR. This confirms that variety (and cardinality) reasoning is highly effective in reducing search space. The extra prunings are so significant that they compensate for the overhead of extra computational effort spent for variety (and cardinality) reasoning. For runtimes, BC+CR+VR is also always the fastest, although the proportion of reduction is less than that for the number of fails. The reduction of BC+CR+VR over BC+CR in Table 4 is moderate, but that in Table 5 is significant. There are even instances in which both BC and BC+CR cannot finish execution within the time limit, but BC+CR+VR can. This also shows that the usefulness of variety reasoning sometimes depends on the tightness of the variety constraints in a problem.

6.3 Generalized social golfer problem

Similar to the extended Steiner system, the generalized social golfer problem extends the standard social golfer problem (prob010 in CSPLib) from set-based to multiset-based. The original social golfer problem requires to schedule m groups of g golfers over w weeks so that no golfer plays in the same group as any other golfer twice. We generalize the problem to an extended version $SG(w, m, n, g, p)$ in which we schedule m teams of n members to g groups of p golfers over w weeks. Each

group contains golfers from different teams and they play against each other. We also constrain each group to have golfers from at least v teams. To maximize the socialisation, the number of times two teams meet with each other again is minimized.

Table 6 shows the experimental results of the generalized social golfer problem and the variety of each multiset is at least 2. Again, BC+CR+VR always has the fewest number of fails and runtime. In this problem, the group size is fixed to p and this favours cardinality reasoning. Thus, the reduction in the number of fails of BC+CR+VR over BC alone is more significant than the reduction of BC+CR+VR over CR+VR. However, with the help of variety reasoning, BC+CR+VR can achieve zero backtrack in some instances. This shows that the extra prunings are so significant and effective in removing inconsistent domain values in the search tree.

6.4 Generalized warehouse location problem

Our last experiment is the generalized warehouse location problem $WH(l, w, s)$. In this problem, we are given a set of locations l where a maximum of w warehouses

Table 6 Results of the generalized social golfer problem

(w, m, n, g, p)	BC		BC+CR		BC+CR+VR	
	Fails	Runtime	Fails	Runtime	Fails	Runtime
(3,2,2,2,2)	2162	0.07	32	0.01	**0**	**0**
(3,2,3,2,2)	42821	1.2	247	0.02	**0**	**0.01**
(3,2,3,2,3)	26025	0.76	71	0.01	**22**	**0**
(3,2,3,3,2)	975	0.04	18	0.01	**0**	**0**
(3,2,4,2,2)	302820	8.16	273	**0.01**	**0**	**0.01**
(3,2,4,2,3)	335774	9.39	528	0.04	**0**	**0**
(3,2,4,2,4)	173955	5.22	97	0.02	**22**	**0**
(3,2,4,3,2)	14334	0.45	234	0.03	**0**	**0.01**
(3,2,4,4,2)	4901736	191.51	345	0.03	**0**	**0**
(3,2,5,2,2)	1343708	35.58	273	0.01	**0**	**0**
(3,2,5,2,3)	1852889	50.06	1301	0.07	**0**	**0.01**
(3,2,5,2,4)	1606845	45.54	1118	0.08	**22**	**0**
(3,2,5,2,5)	788233	23.12	146	0.01	**22**	**0**
(3,2,5,3,2)	99333	3.39	260	0.03	**0**	**0**
(3,2,5,3,3)	227055	8.28	1263	0.1	**0**	**0.01**
(3,2,5,4,2)	–	–	3179	0.23	**0**	**0.01**
(3,2,5,5,2)	957799	45.03	286	0.04	**0**	**0**
(3,3,2,2,2)	1955941	62.01	8818	0.52	**2254**	**0.19**
(3,3,2,2,3)	943173	33.78	369	**0.04**	**336**	0.05
(3,3,2,3,2)	141126	6.6	938	0.12	**444**	**0.09**
(3,3,3,2,2)	43253055	1377.83	25260	1.49	**2422**	**0.21**
(3,3,3,2,3)	–	–	88624	6.98	**45054**	**4.63**
(3,3,3,2,4)	49633275	1754.52	16498	**1.7**	**14934**	**1.7**
(3,3,3,3,2)	23546963	1060.89	176765	17.58	**33467**	**5.08**
(3,3,3,3,3)	23927286	1136.83	36249	5.11	**22423**	**4.19**
(3,3,3,4,2)	–	–	1090365	111.35	**112519**	**23.56**
(3,3,4,2,2)	–	–	25723	1.53	**2461**	**0.19**
(3,3,4,2,3)	–	–	256222	19.99	**94420**	**10.04**

The variety of each multiset is at least 2

can be built to supply s stores. Different stores sell different kinds of goods, which are supplied by different warehouses. A warehouse only produces one kind of goods. There is a supply cost for delivering goods to the stores and this cost depends on the warehouses. Also, there is a cost of 5000 for building a warehouse. This problem is an optimization problem which minimizes the total cost in building the warehouses and the supply costs to the stores.

To model this problem, we use a multiset variable to represent where the warehouses are going to be built and s other multiset variables to represent the stores. The domain values of all the multiset variables are the locations of the warehouses. Union and equality multiset constraints are used to ensure that all stores obtain goods from the warehouses. To better utilize the resource and prevent under production, no two stores can obtain goods from more than v same location. Such a requirement can be modeled by the variety constraints on the intersection of any two stores.

Tables 7 and 8 show the experimental results of the generalized warehouse location problem in which no two stores can obtain goods from more than one and two same locations respectively. Similar to the previous three benchmark

Table 7 Results of the generalized warehouse location problem

(l, w, s)	BC		BC+CR		BC+CR+VR	
	Fails	Runtime	Fails	Runtime	Fails	Runtime
(5, 3, 7)	4749133	237.54	770204	128.27	**244732**	**75.22**
(5, 3, 8)	4855804	277.95	829065	181.75	**274109**	**109.82**
(5, 3, 9)	4880436	316.58	843311	220.99	**281261**	**133.48**
(5, 3, 10)	4889762	354.38	848793	249.4	**284026**	**152.17**
(5, 5, 7)	154002	10.2	9307	2.64	**5554**	**1.7**
(5, 5, 8)	154605	11.6	9315	3.1	**5558**	**2.01**
(5, 5, 9)	155743	11.77	9334	3.48	**5567**	**2.26**
(5, 5, 10)	155993	12.33	9340	3.75	**5572**	**2.43**
(6, 3, 7)	25291006	1403.5	3412505	701.29	**1021670**	**402.47**
(6, 3, 8)	25363951	1608.32	3502747	945.69	**1085141**	**568.89**
(6, 3, 9)	25369950	1789.62	3510070	1151.32	**1090041**	**693.51**
(6, 3, 10)	–	–	3522116	1302.76	**1098070**	**786.23**
(6, 5, 7)	395677	55	24718	8.73	**14509**	**5.33**
(6, 5, 8)	451461	43.7	25028	10.91	**14714**	**6.45**
(6, 5, 9)	451773	52.92	25035	12.26	**14720**	**7.27**
(6, 5, 10)	453102	60.95	25040	13.18	**14726**	**7.85**
(7, 5, 7)	1041329	167.81	56228	25.63	**32493**	**14.36**
(7, 5, 8)	1152650	124.22	56692	31.48	**32796**	**17.18**
(7, 5, 9)	1153222	151.48	56700	35.63	**32803**	**19.55**
(7, 5, 10)	1155232	171.46	56707	38.5	**32810**	**21.01**
(8, 5, 7)	2358226	402.11	113084	65.54	**64483**	**34.1**
(8, 5, 8)	2539984	300.75	113740	79.6	**64908**	**40.72**
(8, 5, 9)	2544059	358.25	113750	90.57	**64915**	**46.33**
(8, 5, 10)	2548600	405.1	113761	98.22	**64923**	**49.9**

The variety of the intersection of any two stores is at most 1

Table 8 Results of the generalized warehouse location problem

(l, w, s)	BC		BC+CR		BC+CR+VR	
	Fails	Runtime	Fails	Runtime	Fails	Runtime
(7, 3, 7)	2286055	385.65	103268	41.95	**97543**	**36.59**
(7, 3, 8)	4611458	845.88	126920	69.31	**118194**	**53.19**
(7, 3, 9)	4736060	1102.1	127875	79.27	**119016**	**60.69**
(7, 3, 10)	5144234	1387.8	130728	89.03	**121439**	**66.57**
(7, 5, 7)	2100474	494.41	95519	38.46	**90678**	**34.47**
(7, 5, 8)	4865139	1368.3	126818	69.13	**118180**	**53.74**
(7, 5, 9)	–	–	127859	79.23	**119076**	**61.32**
(7, 5, 10)	–	–	130814	89.06	**121586**	**67.43**
(8, 3, 7)	6310893	1183.41	220560	113.24	**207944**	**92.45**
(8, 3, 8)	–	–	261425	180.15	**243434**	**130.05**
(8, 3, 9)	–	–	263236	206.26	**244956**	**147.59**
(8, 3, 10)	–	–	267926	231.33	**248926**	**161.56**
(8, 5, 7)	5584477	1583.95	204447	102.16	**193779**	**86.2**
(8, 5, 8)	–	–	260460	177.85	**242826**	**130.35**
(8, 5, 9)	–	–	262379	204.38	**244441**	**148.26**
(8, 5, 10)	–	–	267399	230.23	**248658**	**162.69**

The variety of the intersection of any two stores is at most 2

problems, BC+CR+VR achieves the fewest number of fails and the fastest runtime among the three propagation approaches. The reduction is more significant for the comparison between BC alone and BC+CR+VR than that between BC+CR and BC+CR+VR. When the variety of the intersection of any two stores is limited to at most 2, the problem becomes less constrained even the problem size increases. Thus, the reduction of BC+CR+VR over BC+CR is moderate, but BC+CR+VR still achieves about 25 % faster in runtime.

6.5 Discussions

In the template design problem, extended Steiner problem, and the generalized social golfer problem, the reduction in the number of fails of BC+CR+VR over BC+CR is more significant than that in the runtime. In general, enforcing stronger consistency will discover and remove more inconsistent nodes to give a smaller search tree. However, more computational effort will be spent for constraint propagation at each search node. So, a large number of nodes has to be removed so as to compensate for the increased computational effort for a stronger consistency enforcement and to improve the runtime.

From the experimental results of the generalized warehouse location problem, we observe that the reduction in runtime of BC+CR+VR over BC+VR is greater than that in number of fails. It behaves differently from the previous three benchmarks. So, we did further experiments to investigate the results. We find that the interaction among the multiset equality, union, and intersection constraints in the generalized warehouse location problem provides tighter bounds in BC+CR+VR. The pruning is so significant that they not only compensate for the overhead of the extra com-

putational effort spent for cardinality-variety reasoning, but also reduce the number of iterations for constraint propagation at each search node. Thus, improvement in runtime is even better than that in number of nodes.

Take the generalized warehouse location problem instance $WH(8, 5, 9)$ and the variety of the intersection of any two stores is at most 1 as an example. BC+CR and BC+CR+VR have 113,752 and 64,917 number of choice points (i.e., the number of nodes in the search tree) respectively and the average numbers of iterations for constraint propagation at each search node are 286 and 254 respectively. BC+CR+VR reduces not only the number of choice points, but also the number of iterations it takes to converge at each search node. This results in fewer number of nodes in the search tree and also less work at each node. In contrast, in the extended Steiner system, although BC+CR+VR has fewer number of choice points, there are more iterations at each search node when compared to BC+CR. Take the extended Steiner System instance $ES(2, 4, 5)$ where $b = 5$ and variety of each multiset is at least 2 as an example. BC+CR and BC+CR+VR have 637,200 and 356,786 number of choice points respectively and the average number of iterations for constraint propagation at each search node are 29 and 37 respectively. Thus, improvement in number of nodes is better than that in runtime for the extended Steiner system. As we can see, the multiset constraints can interact with each other and give tighter bounds.

7 Concluding remarks

In this section, we summarize our contributions in this paper and give some possible directions for future research.

7.1 Contributions

Besides integer, Boolean, and set variables and constraints, multiset variables and constraints are also useful in providing more natural and efficient models to many problems. However, we can only find little work on multiset constraint solving. Thus, this paper mainly studies multiset variables and constraints. The contribution of our work can be summarized as follows.

Unlike set variables, multiset variables allow repeated elements. We first propose three different ways to represent a multiset variable: bounds, occurrence, and fixed cardinality representations. The occurrence representation is more expressive than the bounds representation in general, and is as expressive as the bounds representation if we only maintain the bounds on the number of occurrences of an element in a multiset. The fixed cardinality representation is incomparable to the other two representations. Thus, in this paper, we focus on using the occurrence representation.

Second, we have described various inference rules to maintain bounds consistency that work with constraints involving multiset, set, and/or integer variables. This bounds consistency is equivalent to integer bounds consistency applied to the occurrence representation for multiset variables. Those inference rules can tighten the upper and/or lower bound on the multiset variables to give a smaller search space.

Third, we have introduced the cardinality and variety variables to multiset variables based on the occurrence representation. While the introduction of cardinality

variable is a straightforward generalization to the set variable counterpart, the idea of variety variable is a new concept.

Fourth, we have exploited the cardinality and variety properties to introduce new inference rules to increase pruning opportunities. We have also improved the propagation of some common multiset constraints through cardinality and variety reasoning. Experimental results on several benchmark problems confirm that enforcing bounds consistency with cardinality-variety reasoning can always achieve tighter bounds on multiset variables, resulting in an even smaller search space.

CSPs are typically solved by incorporating constraint propagation algorithms in backtracking tree search. Cardinality and variety reasoning is a form of extra constraint propagation. On one hand, it aims at bringing out more domain prunings so that the size of the search tree becomes smaller. On the other hand, it comes at an additional computational cost at each node in the search tree. Striking a balance between the computational overhead and the extra prunings is always problem specific. We can never guarantee that a stronger pruning power can always compensate for the additional computational cost. In our work, the overall effectiveness depends heavily on the existence of cardinality and variety information. From our experimental results, we envisage that if the problem has more constraints on the cardinality and variety of the variables, we can probably benefit to a greater extend.

7.2 Future work

Our work introduces the variety information to multiset variables and constraints. There can be plenty of scope for future research for multiset CSPs.

First, cardinality-variety reasoning is currently applied on some common multiset constraints in their normal forms. It is interesting to study how cardinality-variety reasoning can be incorporated into other multiset global constraints. Since global constraints can be decomposed into their normal form, we can check if the cardinality-variety reasoning on multiset global constraints can achieve even better constraint propagation.

Second, our current implementation of the hybrid representation and the rules is only prototypical. There is still room for improvement. For example, it is known that adjusting the triggering order of the rules (depending on the computational cost of the rules) can affect the performance [1, 2]. We expect that our implementation can be optimized in the future.

Third, we have proposed to use a hybrid representation for multiset variables in this paper. It is worthwhile to study if there are other ways to represent a multiset variable and compare them with the hybrid representation. Besides bounds representation, we can also represent set variables by ordering the set values lexicographically. This is called the length-lex representation [9]. We can investigate how length-lex ordering can be applied to the values in the domain of a multiset variable.

Acknowledgements We would like to thank the anonymous referees for their constructive comments. The work described in this paper was substantially supported by grants (CUHK413207, CUHK413808, and CUHK413710) from the Research Grants Council of Hong Kong SAR. Toby Walsh is supported by the Australian Government's Department of Broadband, Communications and the Digital Economy, the Australian Research Council, and the Asian Office of Aerospace Research and Development through grants AOARD-104123 and 124056.

References

1. Azevedo, F. (2007). Cardinal: a finite sets constraint solver. *Constraints, 12*(1), 93–129.
2. Azevedo, F., & Barahona, P. (2000). Modelling digital circuits problems with set constraints. In *Proceedings of the 1st international conference on computational logic* (pp. 414–428).
3. Bennett, F.E., & Mendelsohn, E. (1980). Extended (2, 4)-designs. *Journal of Combinatorial Theory, Series A, 29*(1), 74–86.
4. Debruyne, R., & Bessière, C. (1997). Some practicable filtering techniques for the constraint satisfaction problem. In *Proceedings of the 15th international joint conference on artificial intelligence* (pp. 412–417).
5. Gecode Team (2006). Gecode: Generic constraint development environment. Available from http://www.gecode.org.
6. Gent, I.P., & Walsh, T. (1999). CSPLib: a benchmark library for constraints. Technical report, Technical report APES-09-1999. A shorter version appears in *Proceedings of the 5th international conference on principles and practices of constraint programming*.
7. Gervet, C. (1994). Conjunto: constraint logic programming with finite set domains. In *Proceedings of the 1994 symposium on logic programming* (pp. 339–358).
8. Gervet, C. (1997). Interval propagation to reason about sets: definition and implementation of a practical language. *Constraints, 1*(3), 191–244.
9. Gervet, C., & Van Hentenryck, P. (2006). Length-lex ordering for set csps. In *Proceedings of the 21st national conference on artificial intelligence* (pp. 48–53).
10. Hawkins, P., Lagoon, V., Stuckey, P.J. (2005). Solving set constraint satisfaction problems using ROBDDs. *Journal of Artificial Intelligence Research, 24*, 109–156.
11. ILOG (2003). *ILOG solver 6.0 reference manual*.
12. Johnson, D.M., & Mendelsohn, N.S. (1972). Extended triple systems. *Aequationes Mathematicae, 8*(3), 291–298.
13. Laburthe, F. (2000). Choco: implementing a CP kernel. In *CP00 post conference workshop on techniques for implementing constraint programming systems (TRICS)* (pp. 71–85).
14. Lagoon, V., & Stuckey, P.J. (2004). Set domain propagation using ROBDDs. In *Proceedings of the 10th international conference on principles and practice of constraint programming* (pp. 347–361).
15. Law, Y.C., Lee, J.H.M., Woo, M.H.C. (2009). Variety reasoning for multiset constraint propagation. In *Proceedings of the 21st international joint conference on artificial intelligence* (pp. 552–558).
16. Mackworth, A.K. (1977). Consistency in networks of relations. *Artificial Intelligence, 8*(1), 99–118.
17. Mohr, R., & Masini, G. (1988). Good old discrete relaxation. In *Proceedings of the 8th European conference on artificial intelligence* (pp. 651–656).
18. Müller, T. (2001). *Constraint propagation in Mozart*. Doctoral dissertation, Universität des Saarlandes, Germany.
19. Müller, T., & Müller, M. (1997). Finite set constraints in Oz. In *13. Workshop logische programmierung* (pp. 104–115).
20. Nadel, B.A. (1989). Constraint satisfaction algorithms. *Computational Intelligence, 5*, 188–224.
21. Park, E.Y., & Blake, I. (2008). Construction of extended Steiner systems for information retrieval. *Revista Matemática Complutense, 21*(1), 179–190.
22. Walsh, T. (2003). Consistency and propagation with multiset constraints: a formal viewpoint. In *Proceedings of the 9th international conference on principles and practice of constraint programming* (pp. 724–738).