Multiset Constraint Solving and Its Applications

多重集約束求解及其應用

Student: Ko-Lung Yuan

Advisor: Jie-Hong Roland Jiang

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Introduction

- Subset-sum problem
  - Given a set of integers and a target integer t, does there exist any non-empty subset whose sum is t?
  - Example:
    - $\mathrm{E} = \{1, 2, 3, 4, 5\}, t = 6$
    - Yes : $\{1, 2, 3\}$ or $\{1, 5\}$ or $\{2, 4\}$
  - NP-complete problem
  - Satisfiability problem / decision problem of arithmetic constraint

Introduction (cont.)

- Many problems have the property of subset-sum problem
  - E.g.
    - K-partition problem
    - Bin-packing problem
    - Knapsack problem
    - Pseudo Boolean constraint (PBC)
    - Symmetry encoding problem (SEP)
    - …
  - But to the best of our knowledge, there does not exist a general method that can easily model these problems

Introduction (cont.)

- Generalization vs. Easy-formulation
  - Boolean satisfiability constraint (SAT)
    - General enough for modeling the decision problems
    - But it is hard to formulate the arithmetic problems into SAT
    - We can not ensure this kind of powerful approach is always efficient
  - The specific approach for particular problem
    - Not feasible for other related problems
    - The effort of modifying them is quite high

Introduction (cont.)

- The conventional subset-sum problem (constraint) has three lacks in generalization
  - Subset-sum problem just focuses on the equality relation of the subset and its corresponding target
    - Knapsack problem

$$\sum(\{1,5\}) = 6$$

  - Limitation of the occurrences of the elements in the given integer set
  - It can not handle the problems with multiple targets
    - k-partition problem

Introduction (cont.)

- The requirements of proposed constraint
  - It must be general and easy to model related problems
  - It must support all kinds of relations including $>, \geq, <, \leq$ and $=$
    - Constraint table
  - It must allow multiple occurrences of the elements
    - Set -> multiset
  - It must can deal with multiple targets.
    - Search-based algorithm

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Multiset

- Multiset is a collection which allows repeat elements
- A (finite) multiset A is a two-tuple (S, μ)
  - $S \subseteq \mathbb{Z}$
  - multiplicity function $\mu(e): \mathbb{Z} \rightarrow \mathbb{N}$
  - $\mu(e)$ denotes how many times an element $e \in \mathbb{Z}$ is present in A
  - for $e \notin S$, we assume $\mu(e) = 0$
  - We simply use $\emptyset$ to present the empty multiset
  - Example:

The same multiset
  - $A = (\{-2, 1, 2, 5\}, \{\mu(-2) = 1,\ \mu(1) = 2,\ \mu(2) = 1,\ \mu(5) = 1\})$
  - $A = \{-2,\ 1,\ 1,\ 2,\ 5\}$

Multiset (cont.)

- Multiset Operation
  - $\sum(A)$: The sum of multiset A
  - $\prod(A)$: The product of multiset A
  - $\mathrm{Max}(A)$: The maximum element in A
  - $\min(A)$: The minimum element in A
  - $|A| = \sum_{e\in S_a}(\mu(e))$: The size of multiset A

  - $A \sqcup B$: The union of multiset $A = (S_a, \mu_a)$ and $B = (S_b, \mu_b)$
    - $A \sqcup B = (S_a \cup S_b, \mu_a + \mu_b)$ with $(\mu_a + \mu_b)(e) = \mu_a(e) + \mu_b(e)$
    - $\sum(A \sqcup B) = \sum(A) + \sum(B)$
    - $\prod(A \sqcup B) = \prod(A) \times \prod(B)$

    $\sum(\emptyset) = 0$ and $\prod(\emptyset) = 1$

  - $A \sqsubseteq B$: $A = (S_a, \mu_a)$ is a sub-multiset of $B = (S_b, \mu_b)$
    - $S_a \subseteq S_b$ and $\mu_a(e) \le \mu_b(e)$ for every $e \in \mathbb{Z}$

Multiset Constraint

- Subset-sum constraint
  - Given two multisets E and $T = \{t^1, \ldots, t^k\}$ of integers, the subset-sum constraint of E summing to T asks whether there exist $E^i \sqsubseteq E$ for $i = 1, \ldots, k$ such that $\sum(E^i) \bowtie t^i$ and $\sqcup E^i \sqsubseteq E$, where $\bowtie \in \{<, \leq, >, \geq, =\}$
- Subset-product constraint
  - Given two multisets E and $T = \{t^1, \ldots, t^k\}$ of integers, the subset-product constraint of E multiplying to T asks whether there exist $E^i \sqsubseteq E$ for $i = 1, \ldots, k$ such that $\prod(E^i) = t^i$ and $\sqcup E^i \sqsubseteq E$

Note: We remove the request that the sub-multiset should be non-empty

Multiset Constraint (cont.)

- The multisets E and T are called element multiset and target multiset, respectively
- We can fix the order of E and T
  - They will become lists called element list and target list, respectively
- The conventional subset-sum problem is the special case with ($\bowtie$ set to $=$) and $|T| = 1$

Multiset Constraint Table

- Constraint Table is a true/false table build by dynamic programming (DP)
  - Example : Given a multiset $E = \{1, 1, 3, 3\}$ and T= $\{1,3,4\}$ for subset-sum constraint with ($\bowtie$ set to $=$)

E \ t	0	1	2	3	4
0	True	False	False	False	False
1	True	True	False	False	False
1	True	True	True	False	False
3	True	True	True	True	True
3	True	$\boxed{\text{True}}$	True	$\boxed{\text{True}}$	$\boxed{\text{True}}$

S[1,4] S[3,4] S[4,4]

$S[i, j]$ ($P[i, j]$):

The value of table entry in the i^{th} row and column indexed by j

Order fixed

Whether we can use first i elements in E to sum (multiply to) j

Multiset Constraint Table (cont.)

- There are three aspects have to be considered for constraint table building
  - Table size
    - Table indexing way
  - Initial values of table
    - Border targets
    - Facts of empty multiset
  - Predicate of table
    - It can compute the values of table entries by the known entry values

decide

It will start from initial values

Multiset Constraint Table (cont.)

- Predicate of subset-sum table
  - $S[i,j] = \underbrace{S[i-1,j]}_{\text{not use } e_i} \vee \underbrace{S[i-1,j-e_i]}_{\text{use } e_i}$ e_i is the i^{th} element in E

An entry references two entries

E \ t	0	1	2	3	4
0	True	False	False	False	False
1	True	True	False	False	False
1	True	True	$\boxed{\text{True}}$	False	False
3	True	True	True	True	True
3	True	True	True	True	True

Multiset Constraint Table (cont.)

- Predicate of subset-product table

- $$P[i,j] = \begin{cases} True, & j = 0 \; and \; e_i = 0 \\ P[i-1,j] \vee P\left[i-1, \frac{j}{e_i}\right], & (j \bmod e_i) = 0 \; and \; e_i \neq 0 \\ P[i-1,j], & others \end{cases}$$

E \ t	-4	-3	-2	-1	0	1	2	3
$\boxed{1}$	False	False	False	False	False	True	False	False
3	False	False	False	False	False	True	$\boxed{\text{False}}$	True
1	False	False	False	False	False	True	False	True
-3	False	$\boxed{\text{True}}$	False	False	False	True	False	$\boxed{\text{True}}$
1	False	True	False	False	False	True	False	True

Multiset Constraint Table (cont.)

- Border targets
  - t_{left} : the left border target is the max integer such that the values of $S[i,j](P[i,j])$ are all true or all false when $j < t_{left}$
  - t_{right} : the right border target is the min integer such that the values of $S[i,j]$ $(P[i,j])$ are all true or all false when $j > t_{right}$

  - That is why we do not have to build an infinite table for all integer targets

Multiset Constraint Table (cont.)

- Example of border targets and boundary conditions

  - Given a multiset $E = \{1, 1, 3, 3\}$ and T= $\{1,3,4\}$ for subset-sum constraint with ($\bowtie$ set to $=$)

  - $S[i, j] = false$ for $j < 0$ t_{left}
  - $S[i, j] = false$ for $j > \sum(E) = 8$ t_{right}

  boundary conditions

  - Note : The summary of the border targets of all kinds of constraints is shown in Table 3.3 and Table 3.8

Multiset Constraint Table (cont.)

- Facts of empty multiset

  - For the case $i = 0$, the values of table entries are known
  - The empty multiset is the only candidate in solution space

  - It can be discussed in three region:
    - $j < 0,\ j = 0$ and $j > 0$ for subset-sum constraint
    - $j < 1,\ j = 1$ and $j > 1$ for subset-product constraint

  $\sum(\emptyset) = 0$ and $\prod(\emptyset) = 1$

Multiset Constraint Table (cont.)

- Example of facts of empty multiset

$E \setminus t$	0	1	2	3	4
0	True	False	False	False	False
1	True	True	False	False	False
1	True	True	True	False	False
3	True	True	True	True	True
3	True	True	True	True	True

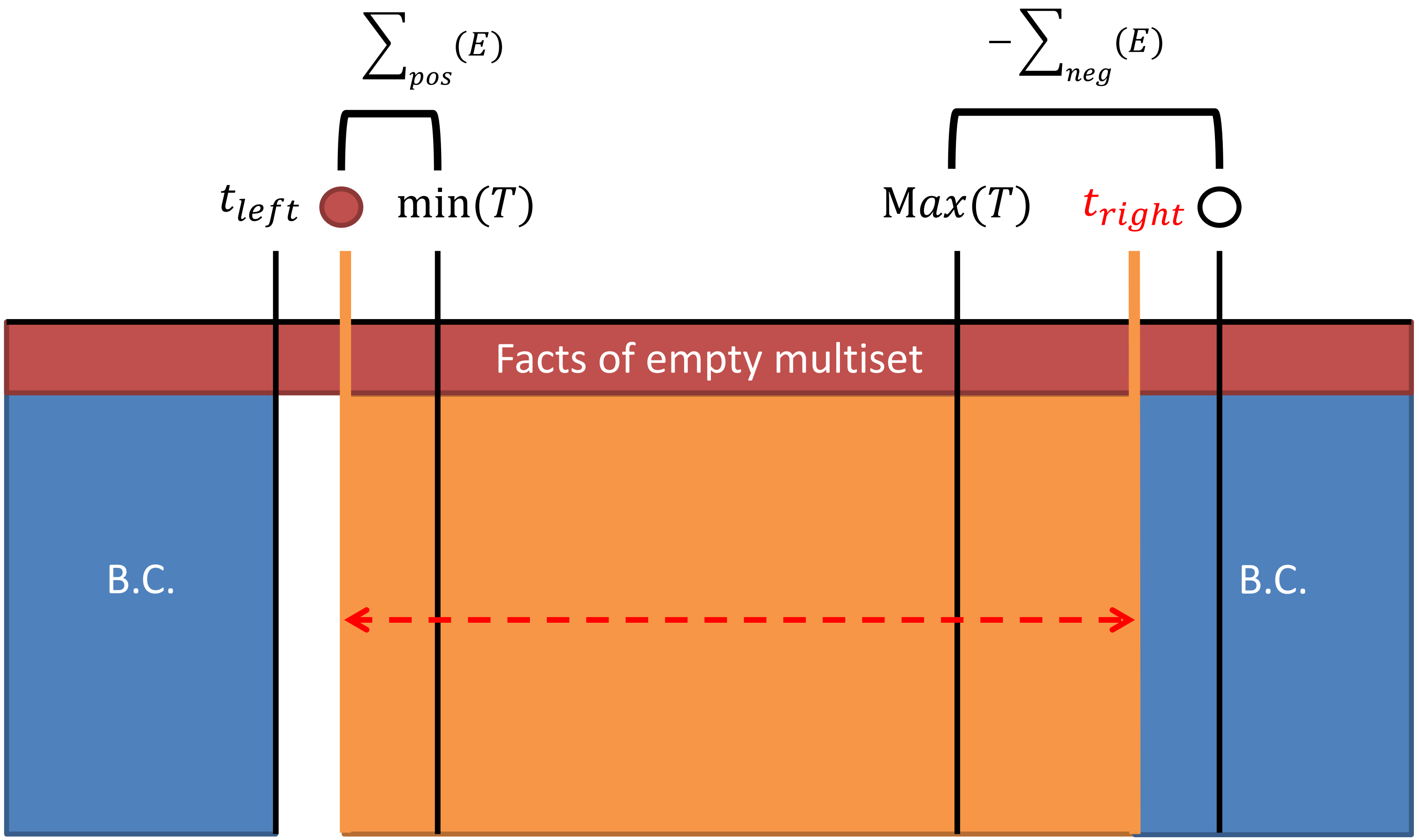
Facts of empty multiset

$$\sum(\emptyset) = 0$$

- Note : The summary of the facts of empty multiset of all kinds of constraints is shown in Table 3.4 and Table 3.9

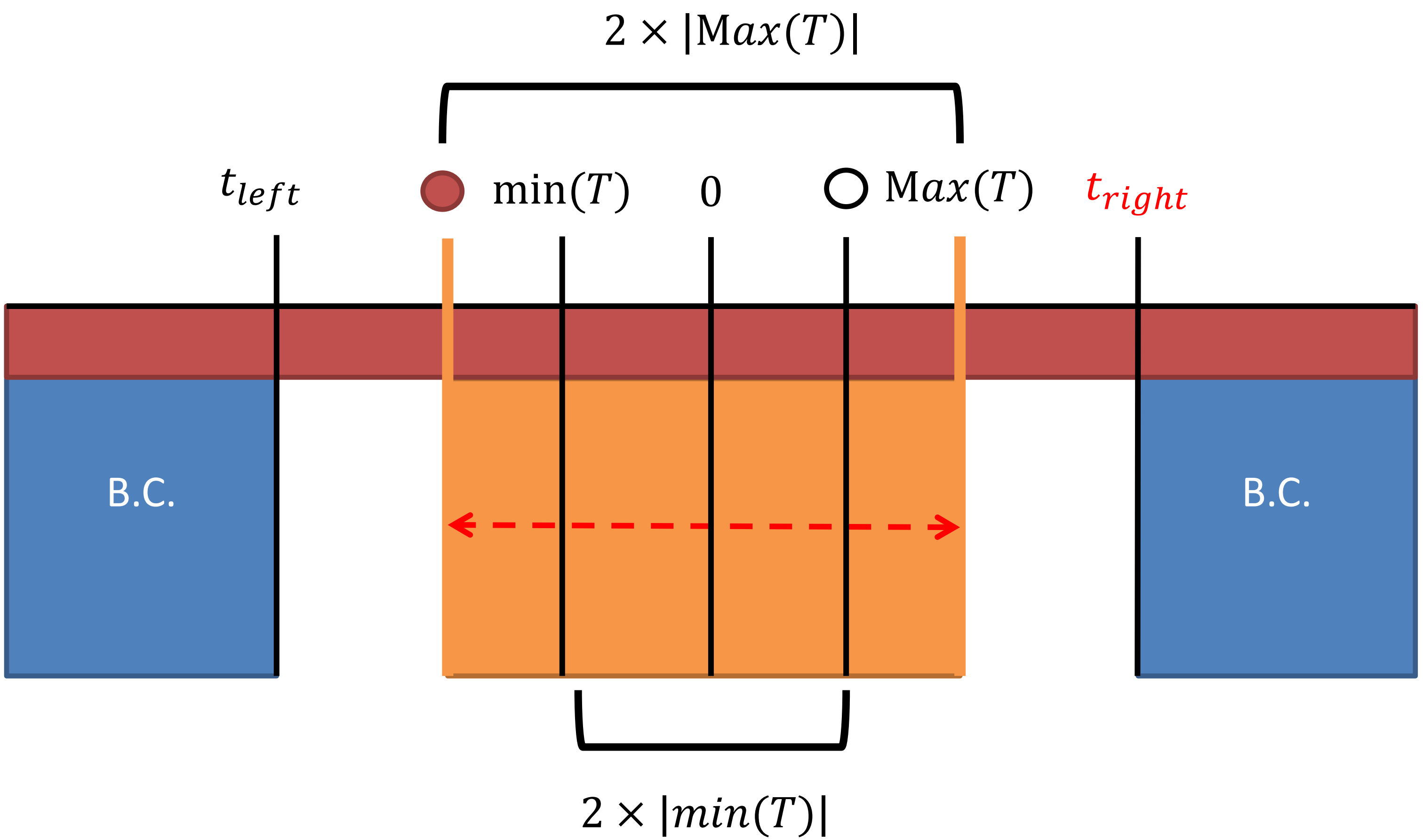
Multiset Constraint Table (cont.)

- Table indexing way of subset-sum constraint



Multiset Constraint Table (cont.)

- Table indexing way of subset-product constraint



Multiset Constraint System

- An object list $O = \{o_1, o_2, \dots, o_n\}$
- An object o_i has k attributes, $1 \leq i \leq n$
- $E^1 = \{e_1^1, e_2^1, \dots, e_n^1\}, \dots, E^k = \{e_1^k, e_2^k, \dots, e_n^k\}$

- A constraint system
  - $MC_1, MC_2, \dots, MC_k$
  - $E^1, E^2, \dots, E^k$
  - If we choose e_i^j from E^j for MC_j, we must choose e_i^x from E^x for MC_x where $1 \leq i \leq n$, $1 \leq j \leq k$ and x from 1 to k element picking rule
  - E.g. knapsack problem

Additional Constraint

- The additional constraints can be seen as the extra requests for original multiset constraint

Constraint	Description
element picking rule	Constraint system
all-use rule	All the elements in element multiset E should be used
must-use rule	Some elements in element multiset E should be used

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Multiset Constraint Solving

- Single target constraint
  - The solution of these single constraint problems can be spotted from the constraint table directly
- Multiple target constraint
  - Search in the constraint table to find a solution that satisfies all targets simultaneously
- Find the sub-multisets
  - Backtracing algorithm

Prepare for Searching

- Find sub-multisets by backtracing [Figure 4.1]
  - Given element list $E = \{1, 1, 3, 3\}$ and target list $\mathrm{T} = \{2, 3, 3\}$ for subset-sum constraint with ($\bowtie$set to $\geq$)

All-true region

	…	-6	…	0	1	2	3
0					False	False	False
1					True	False	False
1					True	True	False
3					True	True	True
3					True	True	True

  - In all-true region, search (backtracing) process will stop its column transition

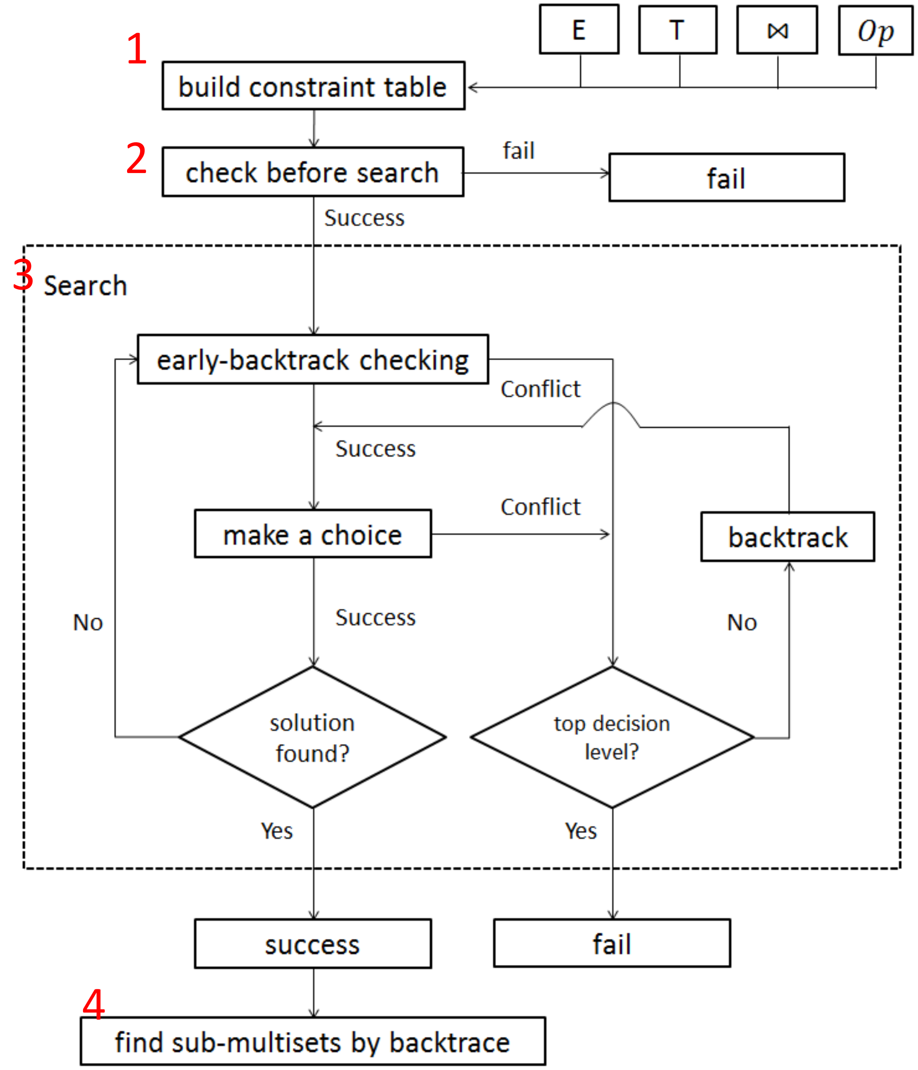
Prepare for Searching (cont.)

- Zero hazard
  - For subset-product constraint, the rows and columns indexed by 0 should be removed since the element in E with value 0 can not be the divisor
  - The way we adopt: remove all the elements with value 0 from element multiset E and target multiset T before constructing constraint table
  - The problem with zero in target multiset T should be separated for treatment [Theorem 4.1 & Proof 4.1]

- The search process has the same concepts and behaviors as backtracing algorithm

Search-based Algorithm

- Algorithm Overview
  - Build constraint table
  - Check before search
  - Search
  - Find sub-multiset by backtracing



Search-based Algorithm (cont.)

- Example
  - Given element list $E = \{1, 1, 3, 3\}$ and target list $\mathrm{T} = \{2, 3, 3\}$ for subset-sum constraint with ($\bowtie$set to $\geq$)
  - Step 1: build constraint table

	-2	-1	0	1	2	3
0	True	True	True	False	False	False
1	True	True	True	True	False	False
1	True	True	True	True	True	False
3	True	True	True	True	True	True
3	True	True	True	True	True	True

Search-based Algorithm (cont.)

- Example
  - Step 2: check before search
    - $S[4,2] = True$
    - $S[4,3] = True$

	-2	-1	0	1	2	3
0	True	True	True	False	False	False
1	True	True	True	True	False	False
1	True	True	True	True	True	False
3	True	True	True	True	True	True
3	True	True	True	True	True	True

Search-based Algorithm (cont.)

- Example
  - Step 3: Search

	-2	-1	0	1	2	3
0	True	True	True	False	False	False
1	True	True	True	True	False	False
1	True	True	True	True	True	False
3	True	True	True	True	True	True
3	True	True	True	True	True	True

choose	candidates
1	
2	
3	
2	3

- False position = 3, 3

All targets are in the all-true region

Search-based Algorithm (cont.)

- Example
  - Step 4: Find sub-multisets by backtracing

	-2	-1	0	1	2	3
0	True	True	True	False	False	False
1	True	True	True	True	False	False
1	True	True	True	True	True	False
3	True	True	True	True	True	True
3	True	True	True	True	True	True

choose	candidates
1	
2	
3	
3	

t^1=2 => $E^1 = \{1, 1\}$
t^1=3 => $E^1 = \{3\}$
t^1=3 => $E^1 = \{3\}$

Improvements

- Prune the search space (use in early backtrack)
  - Check sum
  - Check product
  - Row element using
  - Check distances
  - Learning

- Table simplification

Improvements (cont.)

- Check sum
  - $T(3) = \{1, 4\}$
  - $1 + 4 = 5$
  - Check $S[3, 5]$
  - Suitable for $\bowtie \in \{=, >, \geq, <, \leq\}$
- Check product
  - $T(x) = \{2, 4\}$
  - $2 \times 4 = 8$
  - Check $S[x, 8]$
  - Suitable for $\bowtie \in \{=\}$

E \ T	0	1	2	3	4	5	6
0	True	False	False	False	False	False	False
1	True	True	False	False	False	False	False
3	True	True	False	True	True	False	False
3	True	True	False	True	True	False	True
1	True	True	True	True	True	True	True

Improvements (cont.)

- Row element using
  - $T(4) = \{2, 4\}$
  - $2 + 4 = 6$
  - $e_4 = 1$
  - Check $S[4-1, 6-1]$
  - Suitable for
    - $\bowtie \in \{=\}$
    - subset-sum constraint
    - Subset-product constraint

E \ T	0	1	2	3	4	5	6
0	True	False	False	False	False	False	False
1	True	True	False	False	False	False	False
3	True	True	False	True	True	False	False
3	True	True	False	True	True	False	True
1	True	True	True	True	True	True	True

Improvements (cont.)

- Check distances
  - $T(4) = \{3, 5, 7\}$
  - $d(3) = d(5) = d(7) = 2$
  - Number of $d(t) \leq 2$
    - > 2
    - for t in $T(4)$
  - Backtrack!

  - $T(4) = \{3, 5\}$
  - $d(3) = d = 2$
  - Number of $d(t) \leq 2$
    - $= 2$
    - for t in $T(4)$
  - $e_4 = 1$ and $e_3 = 3$ have to use!

E \ T	0	1	2	3	4	5	6	7	
0	True	False	False	False	False	False	False	False	
2	True	False	True	False	False	False	False	False	
4	True	False	True	False	True	False	True	False	
					2		2		2
3	True	False	True	True	True	True	True	True	
1	True	True	True	True	True	True	True	True	

Improvements (cont.)

- Learning
  - T(2)={2, 2} should backtrack
  - Record {2, 2} in row 2
  - Next time when T(2)={2, 2}
    - Backtrack immediately

E \ t	0	1	2	3	4
0	True	False	False	False	False
1	True	True	False	False	False
1	True	True	True × 2	False	False
3	True	True	True	True	True
3	True	True	True	True	True

Improvements (cont.)

- Summary
  a. Check sum
  b. Check product
  c. Row element using
  d. Check distances
  e. Learning

	Subset-sum	Subset-product
=	a, c, d, e	b, c , e
>	a, d, e	
≥	a, d, e	
<	a, d, e	
≤	a d, e	

Improvements (cont.)

- Table simplification
  - 2-D table becomes 1-D array

E \ t	0	1	2	3	4
0	True	False	False	False	False
1	True	True	False	False	False
1	True	True	True	False	False
3	True	True	True	True	True
3	True	True	True	True	True

→

t	0	1	2	3	4
$A_S[t]$	0	1	2	3	3

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Applications

- k-partition problem [Problem 5.1 & Formulation 5.1]
- Bin-packing problem [Problem 5.2 & Formulation 5.2]
- Knapsack problem [Problem 5.3 & Formulation 5.3]
- Pseudo Boolean constraint solving [Problem 5.4 & Formulation 5.4]
- Symmetry encoding [①] not shown in this slides

Applications (cont.)

- k-partition problem [Problem 5.1 & Formulation 5.1]

  - Example:
    - $S = \{1,\ 2,\ 3,\ 4,\ 5\}$
    - $\sum(S) = 15$
    - 3-parition problem: find three sub-multisets whose sum is $\frac{15}{3} = 5$
    - $S^1 = \{1, 5\},\ S^2 = \{2,\ 3\},\ S^3 = \{5\}$

  - Formulation:
    - Let $E = \{1,\ 2,\ 3,\ 4,\ 5\}$
    - Let $T = \{5,\ 5,\ 5\}$, $\sum(T) = 15$ and $|T| = 3$
    - Subset-sum constraint with ($\bowtie$ set to $=$)

Applications (cont.)

- Bin-packing problem [Problem 5.2 & Formulation 5.2]
  - Example:
    - $S = \{2,\ 2,\ 3,\ 4\}$
    - Bin size $v_b = 5$
    - $N_b = 3 : S^1 = \{2,\ 3\},\ S^2 = \{4\},\ S^3 = \{2\}$
  - Formulation:
    - Let $E = \{2,\ 2,\ 3,\ 4\}$
    - Let $T = \{5\}$ => fail, $T = \{5, 5\}$ => fail , $T = \{5, 5, 5\}$ => succeed
    - Subset-sum constraint with $\bowtie$ set to $\leq$ + all-use rule
    - We could try from $\left\lceil \frac{\sum(S)}{v_b} \right\rceil = \left\lceil \frac{11}{5} \right\rceil = 3$

Applications (cont.)

- Knapsack problem [Problem 5.3 & Formulation 5.3]
  - Example:
    - Object list : $O = \{a, b, c, d, e\}$
    - Value list : $V = \{1,2,1,2,1\}$
    - Size list : $S = \{2,3,1,2,2\}$
    - Knapsack capacity $\mathrm{W} = 5$
    - $A = \{b, d\}$

element picking rule

- Formulation:
  - A constraint system consists two subset-constraints MC_1 and MC_2
  - MC_1 : subset-sum constraint (≤), $E^1 = \{2,3,1,2,2\}$, $T^1 = \{5\}$
  - MC_2 : subset-sum constraint (=), $E^2 = \{1,2,1,2,1\}$, $T^2 = \{V_{max}\}$
  - Try to find the V_{max} (iterative or binary search)

Applications (cont.)

- Pseudo Boolean constraint solving [Problem 5.4 & Formulation 5.4]
  - Example:
    - $3x_1 + 5x_2 - 2x_3 \geq 5$
    - $6x_1 + x_2 + 2x_3 \geq 2$
    - $x_1 = 1, x_2 = 1, x_3 = 0$
  - Formulation:

    element picking rule

    - A constraint system consists two subset-constraints MC_1 and MC_2
    - MC_1 : subset-sum constraint ($\geq$), $E^1 = \{3,5,-2\}, T^1 = \{5\}$
    - MC_2 : subset-sum constraint ($\geq$), $E^2 = \{6,1,2\}, T^2 = \{2\}$
  - Coefficients can be negative, relation can be any other ones

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Experimental Results

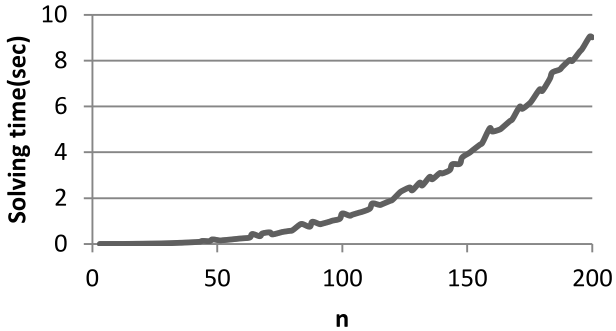
- Multiset constraint solver
  - Implemented as a module in Python(version 3)
- Experimental environment
  - The experiments were conducted on a Linux machine with a Xeon 2.53GHz CPU and 48GB RAM
- Experiments
  - k-partition problem
  - bin-packing problem
  - symmetry encoding [Experimental results of SEP]
  - evaluation of improvements

Experimental Results (cont.)

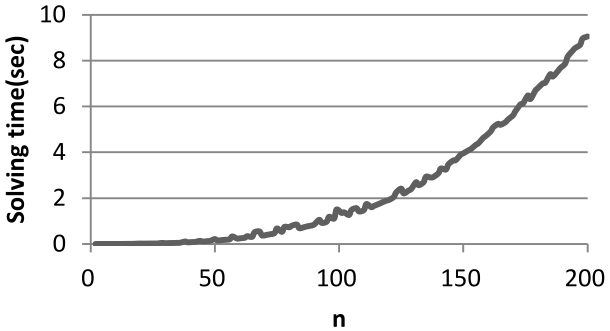
- k-partition problem benchmark
  - For a given integer multiset $S = \{1, \ldots, n\}$, $1 \leq n \leq 200$, if $\sum(S)$ is a multiple of integer k, $2 \leq k \leq 5$, we generate a k-partition problem with multiset S
  - Example:
    - $S = \{1, 2, 3, 4, 5\}$
    - 3-partition problem
    - 5-partition problem

Experimental Results (cont.)

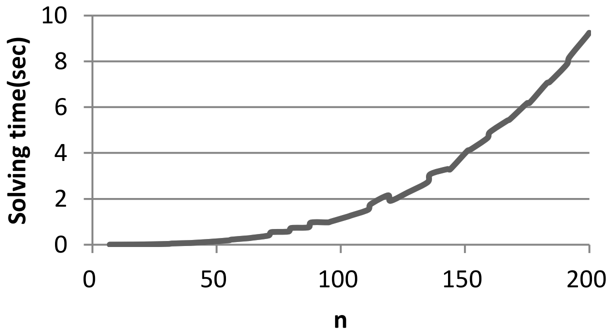
- k-partition problem



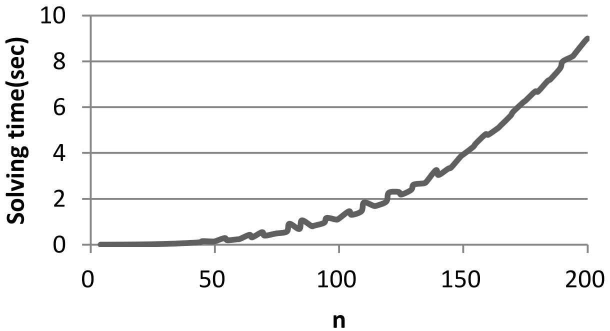
2-partition



3-partition



4-partition



5-partition

Experimental Results (cont.)

- Bin-packing problem benchmark
  - Three parameters
    - number of objects
      - 10, 50, 100, 200, 300, 400
    - maximum size of object (upper bound) v_{max}
      - 10, 50, 100
    - bin size coefficient k
      - 1, 2, 3, 4, 5
      - bin size = $k \times v_{max}$

Experimental Results (cont.)

- Bin-packing problem

#object	Max object size v_{max}	bin size				
		v_{max}	$2 \cdot v_{max}$	$3 \cdot v_{max}$	$4 \cdot v_{max}$	$5 \cdot v_{max}$
10	10	0.001	0.001	0.001	0.001	0.001
	50	0.006	0.003	0.002	0.002	0.003
	100	0.007	0.003	0.003	0.003	0.003
50	10	0.041	0.021	0.009	0.008	0.009
	50	0.110	0.034	0.030	0.029	0.032
	100	9.207	0.060	0.057	0.059	0.061
100	10	0.414	0.040	0.033	0.031	0.029
	50	315.095	0.130	0.124	0.122	0.122
	100	0.786	0.234	0.301	0.227	0.229
200	10	3.678	0.304	0.158	0.220	0.197
	50	$>$3600	0.530	0.491	0.478	0.651
	100	$>$3600	$>$3600	$>$3600	1.046	1.216
300	10	66.795	0.533	0.392	0.334	0.315
	50	14.672	1.649	1.330	1.456	1.357
	100	9.423	2.567	2.428	2.409	2.579
400	10	2.702	1.050	0.917	1.007	0.568
	50	$>$3600	2.807	2.745	2.444	2.516
	100	$>$3600	4.818	4.393	4.463	4.257

Experimental Results (cont.)

- Evaluation of improvements
  - Check sum / check distances / learning
  - Order of element list
  - Benchmark: extract from SEP ($F : \mathbb{Z}_x \rightarrow \mathbb{Z}_4$)
  - Using ratio
    - using ratio of $improvement\ A = \frac{the\ time\ which\ solver\ performs\ backtracking\ due\ to\ A}{total\ backtrack\ time}$
    - The sum of all using ratio: 100%

Experimental Results (cont.)

- Evaluation of improvements

Benchmark	Order	Using ratio(%)				solving time(sec)
		check sum	check distances	learning	others	
mc-32	no	31	11.9	22.6	34.5	0.006
	ascending	69.2	0	12.8	17.9	0.003
	descending	48.7	4.6	22.6	24.1	0.011
mc-64	no	39.5	4.7	31.5	24.3	0.02
	ascending	68.8	0	31.2	0	0.005
	descending	36.3	2.6	38.3	22.7	0.056
mc-128	no	35.2	5.2	32.6	27.1	0.082
	ascending	100	0	0	0	0.008
	descending	34.7	0.3	44.4	20.6	2.38
mc-256	no	37.8	5.4	28.7	28.1	0.275
	ascending	65	0	35	0	0.016
	descending	37	3.8	36.7	22.5	0.289
mc-512	no	35.7	0.6	42.2	21.5	15.260
	ascending	94.7	0	5.3	0	0.02
	descending	33.4	0.2	46.3	20.2	126.262
mc-1024	no	41.8	3.2	27.8	27.2	20.74
	ascending	40.7	0.9	19.3	39.1	0.041
	descending	44.9	9.2	25	20.9	12.909
mc-2048	no	37	1.1	39.4	22.5	327.206
	ascending	50.4	0	29	20.6	0.428
	descending	38.4	0.3	41.1	20.2	135.389
mc-4096	no	-	-	-	-	>3600
	ascending	47.7	0	32.1	20.2	2.256
	descending	-	-	-	-	>3600
mc-8192	no	43.7	2.5	27.7	26	145.34
	ascending	100	0	0	0	0.839
	descending	-	-	-	-	>3600
mc-16384	no	-	-	-	-	>3600
	ascending	52.8	0	25.3	21.9	1.173
	descending	-	-	-	-	>3600

Outline

- Introduction
- Multiset Constraint and Constraint Table
- Multiset Constraint Solving by a Search-based Algorithm
- Applications of Multiset Constraint Solving
- Experimental Results
- Conclusions and Future Work

Conclusions

- A systematic definition of two multiset constraints
  - subset-sum constraint
  - subset-product constraint
- The multiset constraint provides a very powerful expressive ability for modeling many related problems
- Complete method and discussion of building the constraint table

Conclusions (cont.)

- An efficient search-based algorithm with several useful improvements for dealing with the constraint which has multiple targets
- List several important applications which can be easily formulated into multiset constraint and solved efficiently

Future Work

- Constraint table building way of inequality subset-product constraint
  - It is independent of search procedure
- More applications
- Rewrite multiset constraint solver in C++ language
  - For efficiency of solving, native languages have better performance

Thank You for Your Attention

- Q and A

Appendix

- Reference

① K.-L. Yuan, C.-Y. Kuo, J.-H. Jiang, M.-Y. Li. Encoding Multi-Valued Functions for Symmetry. Accepted by International Conference on Computer-Aided Design, 2013.

Appendix – Table 3.3

- Border targets and boundary conditions of subset-sum constraint

E / $Type$	with no negative elements	with negative elements
$\bowtie$ set to $=$	$S[i,j] = \texttt{False}$ for $j < 0$ $S[i,j] = \texttt{False}$ for $j > \sum(E)$	$S[i,j] = \texttt{False}$ for $j < \sum_{neg}(E)$ $S[i,j] = \texttt{False}$ for $j > \sum_{pos}(E)$
$\bowtie$ set to $>$	$S[i,j] = \texttt{True}$ for $j < 0$ $S[i,j] = \texttt{False}$ for $j > \sum(E) - 1$	$S[i,j] = \texttt{True}$ for $j < 0$ $S[i,j] = \texttt{False}$ for $j > \sum_{pos}(E) - 1$
$\bowtie$ set to $\geq$	$S[i,j] = \texttt{True}$ for $j < 1$ $S[i,j] = \texttt{False}$ for $j > \sum(E)$	$S[i,j] = \texttt{True}$ for $j < 1$ $S[i,j] = \texttt{False}$ for $j > \sum_{pos}(E)$
$\bowtie$ set to $<$	$S[i,j] = \texttt{False}$ for $j < 1$ $S[i,j] = \texttt{True}$ for $j > 0$	$S[i,j] = \texttt{False}$ for $j < \sum_{neg}(E) + 1$ $S[i,j] = \texttt{True}$ for $j > 0$
$\bowtie$ set to $\leq$	$S[i,j] = \texttt{False}$ for $j < 0$ $S[i,j] = \texttt{True}$ for $j > -1$	$S[i,j] = \texttt{False}$ for $j < \sum_{neg}(E)$ $S[i,j] = \texttt{True}$ for $j > -1$

Appendix – Table 3.8

- Border targets and boundary conditions of subset-product constraint

E / $Type$	without non-positive elements	with non-positive elements
$\bowtie$ set to $=$	$P[i,j] = \texttt{False}$ for $j < 1$ $P[i,j] = \texttt{False}$ for $j > \prod(E)$	$P[i,j] = \texttt{False}$ for $j < \prod_{min}(E)$ $P[i,j] = \texttt{False}$ for $j > \prod_{Max}(E)$

Table 3.8: Border targets and boundary conditions of subset-product constraint

Appendix – Table 3.4

- Facts of empty multiset of subset-sum constraint

Region / *Type*	$j < 0$	$j = 0$	$j > 0$
$\bowtie$ set to $=$	$S[0, j]$ =False	$S[0, j]$ =True	$S[0, j]$ =False
$\bowtie$ set to $>$	$S[0, j]$ =True	$S[0, j]$ =False	$S[0, j]$ =False
$\bowtie$ set to $\geq$	$S[0, j]$ =True	$S[0, j]$ =True	$S[0, j]$ =False
$\bowtie$ set to $<$	$S[0, j]$ =False	$S[0, j]$ =False	$S[0, j]$ =True
$\bowtie$ set to $\leq$	$S[0, j]$ =False	$S[0, j]$ =True	$S[0, j]$ =True

Appendix – Table 3.9

- Facts of empty multiset of subset-product constraint

Region / *Type*	$j < 1$	$j = 1$	$j > 1$
$\bowtie$ set to $=$	$P[0, j] =$False	$P[0, j] =$True	$P[0, j] =$False

Appendix – Figure 4.1

- Algorithm: find sub-multisets b by backtracing

FindSubMultiset

input: element list E, target t, constraint table $Table$

output: sub-multiset(of E) $SubMultiset$

begin

$i := |E|$;

$j := t$;

$SubMultiset := \emptyset$;

if $S[i, j]$ =`False`

    return `Fail`;

while $i > 0$

    if $S[i-1, j]$ =`False`

        $SubMultiset \sqcup \{E[i]\}$;

        $j := j - E[i]$;

    $i := i - 1$;

return $SubMultiset$;

end

Appendix – Problem 5.1 & Formulation 5.1

- k-partition problem

Problem 5.1 (k-partition Problem) Given a multiset of integers S, can S be partitioned into k sub-multisets $S^1, S^2, \ldots, S^k$ such that the summation of the numbers in each sub-multiset is equal, i.e., $\sum(S^1) = \sum(S^2) =, \ldots, \sum(S^k) = \sum(S)/k$. The sub-multisets $S^1, S^2, \ldots, S^k$ must be disjoint in S and cover S that is $S^1 \sqcup S^2 \sqcup \ldots \sqcup S^k = S$ and $\sum(S)$ must be the multiple of integer k.

Formulation 5.1 Let element multiset $E = S$ and target multiset $T = \{t_i | t_i = \sum(S)/k, 1 \le i \le k\}$ for subset-sum constraint with $\bowtie$ set to $=$. If this constraint is satisfiable, the multiset S can be partitioned successfully.

Appendix – Problem 5.2 & Formulation 5.2

- Bin-packing problem

Problem 5.2 (Bin-packing Problem) Given a bin B with size v_b and n items with sizes $v_1, \ldots, v_n$ to pack, find the smallest number of bins N_b and a N_b-partition $S^1, S^2, \ldots, S^{N_b}$ of the multiset $\{v_1, \ldots, v_n\}$ such that $\sum(S^k) \leq v_b$ for all $k = 1, \ldots, N_b$. Note that the size v_b must be greater than or equal to the maximum size of the items.

Formulation 5.2 Let element multiset $E = \{v_1, \ldots, v_n\}$ and target multiset $T = \{t_i | t_i = v_b, 1 \leq i \leq N_b\}$ for subset-sum constraint with $\bowtie$ set to $\leq$. And we request all the elements in E should be used.(For multiset constraint solving, it is an additional constraint.) The smallest positive integer N_b makes the constraint satisfied is the integer we want to find.

Appendix – Problem 5.3 & Formulation 5.3

- Knapsack problem

Problem 5.3 (0-1 Knapsack Problem) Let there be n objects, $o_1, o_2, \ldots, o_n$ where o_i has a value v_i and weight w_i. The maximum capability of the knapsack is W. The 0-1 knapsack problem is to maximize the sum of the values of the objects in the knapsack but the sum of the weights must be less than capability W. Commonly, we will assume that the values and weights are all non-negative.

Formulation 5.3 Let us build a constraint system which includes two multiset constraint MC_1 and MC_2 with object list $O = \{o_1, o_2, \ldots, o_n\}$. The element list and target list of MC_1 are $E^1 = \{w_1, w_2, \ldots, w_n\}$ and $T^1 = \{W\}$ respectively. MC_1 represents the constraint that the weights must be less than W so it is a subset-sum constraint with $\bowtie$ set to $\leq$. The element list and target list of MC_2 are $E^2 = \{v_1, v_2, \ldots, v_n\}$ and $T^2 = \{V_{max}\}$ respectively where $0 \leq V_{max} \leq \sum(E^2)$. MC_2 is a subset-sum constraint with $\bowtie$ set to $=$. The maximum value of V_{max} makes the constraint system satisfied is the value we want to find.

Appendix – Problem 5.4 & Formulation 5.4

- Pseudo Boolean constraint solving

Problem 5.4 (Pseudo Boolean Constraint System) Linear pseudo Boolean constraint(LPB) is an inequality of the form $\sum_{i=1}^{n} a_i \cdot x_i \geq k$ where $a_i, k \in \mathbb{Z}$ and $x_i \in \{0,1\}$. Variable x_i is a Boolean variable and $\{x_1, x_2, \ldots, x_n\}$ is a Boolean variable set. The integer k and integer multiset $\{a_1, a_2, \ldots, a_n\}$ are called degree and coefficients of LPB respectively. LPB is satisfiable if and only if there is a truth assignment of Boolean variable set such that the inequality is fulfilled. Pseudo Boolean constraint system consists of m constraints LPB_j, $j = 1, 2, \ldots, m$ with Boolean variable set $\{x_1, x_2, \ldots, x_n\}$. And LPB_j has its coefficients $A_j = \{a_1^j, a_2^j, \ldots, a_n^j\}$ and degree k_j. The system is satisfiable if and only if there is a truth assignment of Boolean variable set such that all PBC are satisfied.

Formulation 5.4 Let us build a constraint system includes m multiset constraints MC_j, $j = 1, 2, \ldots, m$ and fix the order of the Boolean variable set. So the coefficients A_j becomes a list with a specific order corresponding to the Boolean variable set. The element list and target list of MC_j are $E = A_j$ and $T = \{k_j\}$ respectively. All of the constraints are subset-sum constraint with $\bowtie$ set to $\geq$. The pseudo Boolean constraint system can be satisfied if and only if the multiset constraint system can be satisfied.

Appendix – Theorem 4.1 & Proof 4.1

- Zero hazard handling

Theorem 4.1 Given a subset-product constraint SPC_1 with element multiset E and target multiset T. Let the number of zero in E and T be N_0^E and N_0^T respectively. If $N_0^E \geq N_0^T$ and another subset-product constraint SPC_2 whose element multiset and target multiset are E and T_{no_zero} is satisfiable, SPC_1 is satisfiable.

Proof 4.1 Let $T = T_0 \sqcup T_{no\_zero}$. If SPC_2 is satisfiable, all targets in T_{no_zero} can be satisfied simultaneously under constraint SPC_1. Also, $N_0^E \geq N_0^T$. It indicates that each target in T_0 can be satisfied by picking a zero from E as one member of the corresponding sub-multiset so the products of these corresponding sub-multisets are all zero.

Appendix – Experimental Results

- Symmetry encoding benchmark
  - Random generated multi-valued function
    - $F : \mathbb{Z}_x \rightarrow \mathbb{Z}_4$
      - $x = 2^5, \dots, 2^{14}$
    - $F : \mathbb{Z}_x \times \mathbb{Z}_x \rightarrow \mathbb{Z}_2$
      - $x = 10, \dots, 50$
    - $F : \mathbb{Z}_x \times \mathbb{Z}_x \rightarrow \mathbb{Z}_2 \times \mathbb{Z}_2$
      - $x = 10, \dots, 50$
    - Ten functions are generated for each case
  - State encoding
    - The FSM examples of the LGSynth89 benchmark suite

Appendix – Experimental Results (cont.)

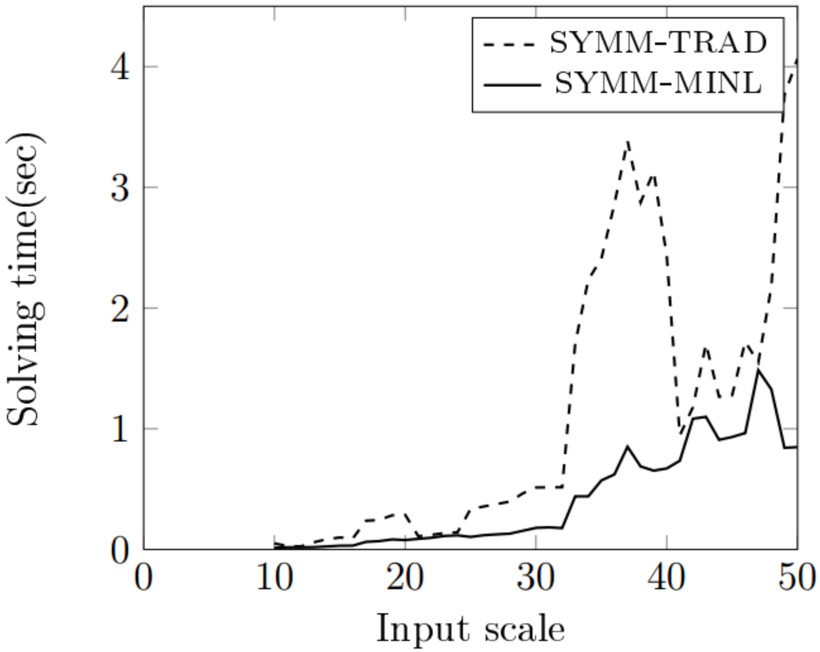
- Symmetry encoding $F: \mathbb{Z}_x \rightarrow \mathbb{Z}_4$

Benchmark	SYMM-MAXS			SYMM-TRAD			SYMM-MINL		
	#inbit	#outbit	time	#inbit	#outbit	time	#inbit	#outbit	time
random-32	5.9	2	0	5.9	2	0.01	5	2	0
random-64	7	2	0.01	7	2	0.02	6	2	0.01
random-128	8	2	0.04	8	2	0.06	7	2	0.08
random-256	9	2	0.03	9	2	0.06	8	2	0.05
random-512	10	2	0.05	10	2	0.27	9	2	0.21
random-1024	11	2	0.12	11	2	0.37	10	2	0.29
random-2048	12	2	0.18	12	2	0.49	11	2	0.34
random-4096	13	2	0.39	13	2	1.05	12	2	0.78
random-8192	14	2	0.66	14	2	6.12	13	2	6.97
random-16384	15	2	5.17	15	2	66.41	14	2	94.81

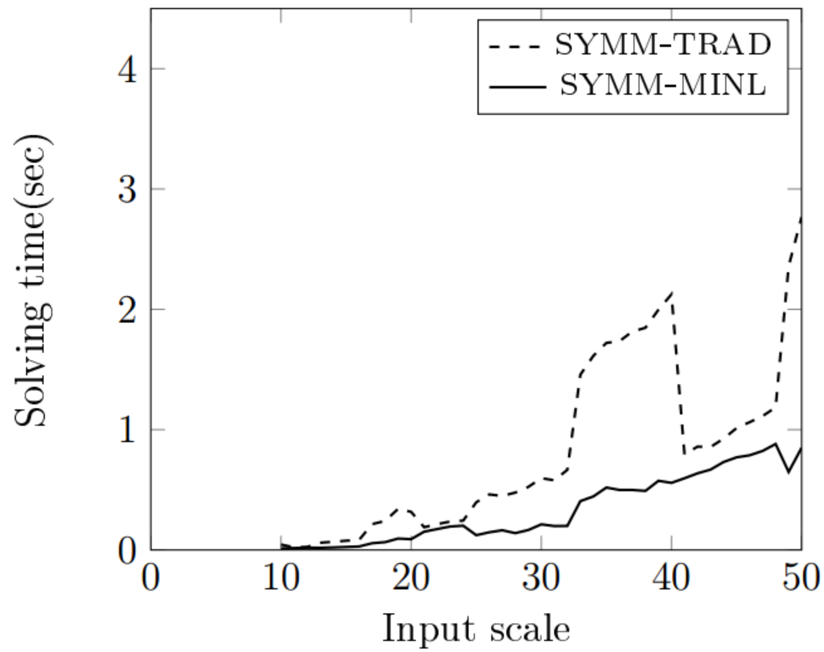
Appendix – Experimental Results (cont.)

- Symmetry encoding

$\mathbb{Z}_x \times \mathbb{Z}_x \to \mathbb{Z}_2$



$\mathbb{Z}_x \times \mathbb{Z}_x \to \mathbb{Z}_2 \times \mathbb{Z}_2$



Appendix – Experimental Results (cont.)

- Symmetry encoding
  - LGSynth89 benchmark suite

Name	#in	#out	#state	SYMM-TRAD			SYMM-MINL		
				sb	ratio	time	sb	ratio	time
bbara	4	2	10	5	19.44	0.12	4	7.14	0.06
bbsse	7	7	16	5	4.55	1.63	4	0	0.75
bbtas	2	2	6	3	10.00	0.00	3	10.00	0.00
beecount	3	4	7	4	14.29	0.03	3	0	0.01
cse	7	7	16	5	6.06	1.71	4	1.82	0.77
dk14	3	5	7	4	14.29	0.03	3	0	0.01
dk15	3	5	4	3	20.00	0.01	2	0	0.01
dk16	2	3	27	6	10.71	0.07	5	0	0.02
dk17	2	3	8	4	20.00	0.01	3	0	0.01
dk27	1	2	7	4	30.00	0.01	3	0	0.00
dk512	1	3	15	5	20.00	0.03	4	0	0.01
donfile	2	1	24	5	4.76	0.03	5	4.76	0.02
ex1	9	19	20	6	5.71	177.88	5	1.10	26.70
ex2	2	2	19	6	21.43	0.04	5	4.76	0.02
ex3	2	2	10	5	28.57	0.02	4	6.67	0.01
ex4	6	9	14	5	5.45	0.59	4	0	0.43
ex5	2	2	9	5	28.57	0.02	4	6.67	0.01
ex6	5	8	8	4	8.33	0.12	3	0	0.08
ex7	2	2	10	5	28.57	0.03	4	6.67	0.01
keyb	7	2	19	6	7.69	2.99	5	1.52	1.19
lion	2	1	4	3	28.57	0.00	2	6.67	0.00
lion9	2	1	9	5	30.00	0.02	4	0	0.01
mc	3	5	4	3	20.00	0.01	2	0	0.01
modulo12	1	1	12	4	10.00	0.01	4	10.00	0.00
planet	7	19	48	6	1.28	9.65	6	1.28	9.64
s1	8	6	20	6	6.59	13.89	5	1.28	4.12
s8	4	1	5	4	21.43	0.03	3	4.76	0.01
shiftreg	1	1	8	4	30.00	0.01	3	0	0.00
sse	7	7	16	5	4.55	1.59	4	0	0.73
styr	9	10	30	6	2.86	80.28	5	0	21.38
tav	4	4	4	3	14.29	0.02	2	0	0.01
tbk	6	3	32	6	4.55	1.47	5	0	0.72
train11	2	1	11	5	28.57	0.03	4	6.67	0.01
train4	2	1	4	3	40.00	0.01	2	16.67	0.00