Improvements of Symmetry Breaking During Search

Zichen Zhu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
zzhu@cse.cuhk.edu.hk

Abstract

Symmetries are common in many constraint problems. They can be broken statically or dynamically. The focus of this paper is the symmetry breaking during search (SBDS) method that adds conditional symmetry breaking constraints upon each backtracking during search. To trade completeness for efficiency, partial SBDS (ParSBDS) is proposed by posting only a subset of symmetries. We propose an adaptation method recursive SBDS (ReSBDS) of ParSBDS which extends ParSBDS to break more symmetry compositions. We observe that the symmetry breaking constraints added for each symmetry at a search node are nogoods and increasing. A global constraint (incNGs), which is logically equivalent to a set of increasing nogoods, is derived. To further trade pruning power for efficiency, we propose weak-nogood consistency (WNC) for nogoods and a lazy propagator for SBDS (and its variants) using watched literal technology. We further define generalized weak-incNGs consistency (GWIC) for a conjunction of increasing nogoods, and give a lazy propagator for incNGs.

Recursive SBDS

SBDS[Gent and Smith, 2000] leaves no symmetric solutions since all symmetries are given. For problems with exponential number of symmetries, direct use of SBDS is impractical [Gent and Smith, 2000]. To trade completeness for efficiency, partial SBDS (ParSBDS) is proposed by posting only a subset of symmetries [Petrie and Smith, 2003].

We prove that ParSBDS has the pitfall to leave more symmetric subtrees and solutions than the widely used static method LexLeader [Crawford et al., 1996] given the same subset of symmetries and the input-order variable and minimum (maximum) value heuristics. We propose Recursive SBDS (ReSBDS) [Lee and Zhu, 2014a] to circumvent the pitfall of ParSBDS. The main idea of ReSBDS is to add extra symmetry breaking constraints during search recursively to prune also symmetric nodes of some pruned subtrees. Thus, ReSBDS can break extra symmetry compositions. When given generators of interchangeable variables (values) according to a static search heuristic, ReSBDS is complete in eliminating the entire symmetry group. We propose further a light version of ReSBDS method (LReSBDS) [Lee and Zhu, 2014b], which has similar pruning power of ReSBDS but with a reduced overhead.

An Increasing-Nogoods Global Constraint

An overhead for SBDS and its variants is the addition of a large number of nogoods with weak pruning power. We observe that the symmetry breaking constraints added for each symmetry at a search node are nogoods that are semantically related. We propose the notion of increasing nogoods [Lee and Zhu, 2014b]. A global constraint (incNGs) [Lee and Zhu, 2014b], which is logically equivalent to a set of increasing nogoods, is derived. Thus we can maintain only one incNGs for each given symmetry. A polynomial time filtering algorithm for incNGs and also an incremental version are proposed.

Weak-Nogood Consistency

Nogoods are weak in pruning and maintaining GAC is not cost effective even when the two watched literal technique [Moskewicz et al., 2001] is utilized. We propose weak-nogood consistency (WNC) [Lee and Zhu, 2015], a weaker consistency notion for nogoods to trade pruning power for efficiency. We propose an efficient lazy propagator to enforce WNC for SBDS (and its variants) using one watched literal [Lee and Zhu, 2015]. First, our propagator generates the watched literal (symmetric assignment) on demand. Second, the propagator is triggered lazily when the watched literal becomes true, and effects prunings only when all but the last assignment of the nogood is true.

A similar weaker consistency, generalized weak-incNGs consistency (GWIC) [Lee and Zhu, 2015], together with a lazy propagator is also proposed for the incNGs global constraint [Lee and Zhu, 2015]. GWIC on a conjunction is equivalent to WNC on individual nogoods. By exploiting the increasing property of the nogoods in incNGs, our lazy propagator watches also one assignment for each global constraint, and operates and benefits from a similar lazy principle.

Experimental Results

This section gives one experiment on Balanced Incomplete Block Design (BIBD) to break matrix symmetries (variable symmetries). A BIBD instance can be determined by its

Table 1: BIBD with Maximum Value Ordering (all solutions)

v, k, λ	**DoubleLex**			**LexLeader**		
	$\#s$	$\#f$	t	$\#s$	$\#f$	t
7,3,5	33,304	191,223	**2.12**	**5,979**	**41,978**	65.64
7,3,6	250,878	1,814,425	21.06	**33,824**	**292,634**	172.44
7,3,7	1,460,332	13,149,270	154.79	**203,296**	**2,069,840**	611.51
7,3,8	6,941,124	76,463,115	886.95	-	-	-
8,4,6	2,058,523	14,156,697	157.75	**596,399**	3,873,360	**118.12**

ParSBDS$_{GAC}$			**ParSBDS**$_{incNGs}$			**ParSBDS**$_{WNC/GWIC}$			
$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t_W	t_G
12,936	83,578	34.95	12,936	83,578	7.25	7,916	54,608	2.39	4.84
93,713	717,959	377.91	93,713	717,959	44.37	41,388	353,232	18.07	20.50
476,752	4,486,587	3,349.82	476,752	4,486,587	270.15	226,176	2,292,110	137.22	114.92
305,312	3,583,192	3,600.00	-	-	-	1,134,253	13,599,864	-	694.02
932,022	6,450,151	2,366.52	932,022	6,450,183	281.22	925,504	6,483,468	351.32	177.09

LReSBDS$_{GAC}$			**LReSBDS**$_{incNGs}$			**LReSBDS**$_{WNC/GWIC}$			
$\#s$	$\#f$	t	$\#s$	$\#f$	t	$\#s$	$\#f$	t_W	t_G
5,979	**41,978**	13.24	**5,979**	**41,978**	5.46	**5,979**	**41,978**	2.89	4.20
33,824	**292,634**	152.99	**33,824**	**292,634**	24.58	**33,824**	**292,634**	25.14	**18.37**
-	-	-	**203,296**	**2,069,840**	145.44	**203,296**	**2,069,840**	224.6	**111.09**
-	-	-	**1,075,694**	**12,921,639**	927.34	**1,075,694**	**12,921,639**	-	**654.21**
596,399	**3,873,339**	445.98	**596,399**	**3,873,339**	169.36	**596,399**	3,956,200	287.32	129.06

parameters (v, k, λ). We use the same model by Lee and Zhu [2014b]. We first solve the benchmarks using the efficient and widely used static method Doublelex [Flener et al., 2002], and also LexLeader to break a much larger subset of symmetries. We then report the results of two dynamic methods ParSBDS and LReSBDS. ReSBDS are discarded in the comparison since LReSBDS is substantially more efficient [Lee and Zhu, 2014b] than it. Each dynamic method would be implemented with the four propagators: GAC on each nogood (GAC), the filtering algorithm of incNGs given by Lee and Zhu [2014b] ($incNGs$), WNC on each nogood (WNC) and GWIC on each incNGs constraint ($GWIC$). ParSBDS is given any two rows or columns being permutable and the Cartesian products of these two subsets. LReSBDS and LexLeader are both given adjacent rows or columns being permutable and the Cartesian products of any two rows or columns being permutable.

All experiments are conducted using Gecode Solver 4.2.0 on Xeon E5620 2.4GHz processors with 7GB. In the table, $\#s$ denotes the number of solutions, $\#f$ denotes the number of failures and t denotes the runtimes. Since WNC and GWIC have the same pruning power, we show their solutions and failures together and use t_W and t_G to denote the runtime of WNC and GWIC respectively. The search time out limit is 1 hour. An entry with the symbol "$-$" indicates that memory is exhausted. The best results are highlighted in **bold**. We use input variable ordering and maximum value ordering. DoubleLex orders rows and columns decreasingly.

Table 1 shows the results. The solution and search tree size by using LReSBDS$_{GAC}$ are reduced to half compared with that of ParSBDS$_{GAC}$. The time is improved 3.47 times on average. After introducing incNGs constraint, ParSBDS and LReSBDS are improved 8.54 and 3.76 times on runtimes on average respectively. For our lazy propagators, ParSBDS$_{WNC}$ and LReSBDS$_{WNC}$ run 16.67 and 4.07 times faster than ParSBDS$_{GAC}$ and LReSBDS$_{GAC}$ on average respectively. ParSBDS$_{GWIC}$ and LReSBDS$_{GWIC}$ run 1.90 and 1.34 times faster than ParSBDS$_{incNGs}$ and LReSBDS$_{incNGs}$ on average respectively. LReSBDS$_{GWIC}$ performs the best and runs 7.90 and 1.12 times faster than LexLeader and DoubleLex on average respectively. Dynamic method beats static method. The benefits of our proposed three improvements are thus demonstrated.

Future Work

For partial symmetry breaking, McDonald and Smith (2006) show there may be a point where the benefit in reducing search from adding more symmetries is out-weighed by the extra overhead. To find the optimum point which has the minimum total runtime, we do not only need to find the number of symmetries to post but also which subset to choose. It is worthwhile to find a good subset of symmetries to break for static and dynamic partial symmetry breaking methods.

McDonald and Smith (2006) have shown how dynamic methods are affected by variable and value heuristics in partial symmetry breaking. The ReSBDS and LReSBDS method are no exception. It is worthwhile to investigate the interaction between these partial methods and search heuristics.

Nogood learning is a general technique for improving backtracking search [Dechter, 1990]. We envision that the increasing-nogoods constraint and the weak-nogood consistency are applicable to other scenarios in CP and SAT, in addition to symmetry breaking.

References

[Crawford et al., 1996] Crawford, J.; Ginsberg, M.; Luks, E.; and Roy, A. 1996. Symmetry breaking predicates for search problems. In *KR'96*, 148–159.

[Dechter, 1990] Dechter, R. 1990. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence* 273–312.

[Flener et al., 2002] Flener, P.; Frisch, A.; Hnich, B.; Kiziltan, Z.; Miguel, I.; Pearson, J.; and Walsh, T. 2002. Breaking row and column symmetries in matrix models. In *CP'02*, 187–192.

[Gent and Smith, 2000] Gent, I., and Smith, B. 2000. Symmetry breaking in constraint programming. In *ECAI'00*, 599–603.

[Lee and Zhu, 2014a] Lee, J., and Zhu, Z. 2014a. Boosting SBDS for partial symmetry breaking in constraint programming. In *AAAI'14*, 2695–2702.

[Lee and Zhu, 2014b] Lee, J., and Zhu, Z. 2014b. An increasing-nogoods global constraint for symmetry breaking during search. In *CP'14*, 465–480.

[Lee and Zhu, 2015] Lee, J., and Zhu, Z. 2015. Filtering nogoods lazily in dynamic symmetry breaking during search. In *IJCAI'15*.

[McDonald and Smith, 2006] McDonald, I., and Smith, B. 2006. Partial symmetry breaking. In *CP'06*, 207–213.

[Moskewicz et al., 2001] Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient sat solver. In *DAC'01*, 530–535.

[Petrie and Smith, 2003] Petrie, K. E., and Smith, B. M. 2003. Symmetry breaking in graceful graphs. In *CP'03*, 930–934.