Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Accelerating SAT Solving by Common Subclause Elimination

Yaowei Yan[1], **Chris E. Gutierrez**[2], **Jeriah Jn-Charles**[2], **Forrest S. Bao**[1] **and Yuanlin Zhang**[2]
[1] Dept. of Electrical & Computer Engineering, University of Akron, Akron, OH, USA
[2] Dept. of Computer Science, Texas Tech University, Lubbock, TX, USA
{yy28, fbao5}@uakron.edu, {chris.e.gutierrez, y.zhang}@ttu.edu, jjncharles@hotmail.com

Abstract

Boolean SATisfiability (SAT) is an important problem in AI. SAT solvers have been effectively used in important industrial applications including automated planning and verification. In this paper, we present novel algorithms for fast SAT solving by employing two common subclause elimination (CSE) approaches. Our motivation is that modern SAT solving techniques can be more efficient on CSE-processed instances. Empirical study shows that CSE can significantly speed up SAT solving.

Introduction

Boolean SATisfiability (SAT) is a well-known and intensively studied NP-complete problem in Computer Science and Artificial Intelligence. A SAT problem is a set of clauses each of which is a set of literals. A subclause is a subset or subsequence of a clause. In this paper, two subclauses (from distinct clauses) are called *common subclauses* if they are equal (as sets or as sequences). Common subclauses are not rare. For instance, more than 90% of problems in SAT Competition 2013 contain common subclauses (Table 2).

Inspired by common subexpression elimination research in programming languages, we mean to replace all occurrences of the common subclause by a new variable, and introduce new clauses to represent the equivalence between the new variable and the common subclause. We assume that CSE likely has a positive impact on SAT solving, taking the advantage of SAT solving techniques such as unit propagation (Katsirelos et al. 2013). Consider this SAT problem: $\{\{a, b, c\}, \{a, b, d\}\}$, where each letter represents a Boolean variable. Replacing the common subclause $\{a, b\}$ with a new variable e, we get new clauses $\{\{e, c\}, \{e, d\}\}$ and those resulting from $e \leftrightarrow a \vee b$: $\{e, \neg a\}$, $\{e, \neg b\}$ and $\{\neg e, a, b\}$. If e is true then there is only one simplified clause $\{a, b\}$ left, and similarly when e is false.

In the following of this paper, we propose using CSE in SAT solving and empirically evaluate how it affects the efficiency of a SAT solver.

CSE Algorithms

We propose two approaches to CSE: frequency based and LZW based.

Frequency Based CSE

The frequency based CSE is to eliminate the most frequent common subclauses first and repeat this process until there is no common subclause more frequent than a preset threshold. For example, consider clauses $\{\{a, b, c\}, \{a, b, c, d\}, \{a, b, e\}\}$. $\{a, b\}$ and $\{b, c\}$ are both common subclauses but the former is the most frequent one and hence is to be replaced first. After eliminating common subclause $\{a, b\}$ with new variable f, we have new clauses $\{\{f, c\}, \{f, c, d\}, \{f, e\}\}$ and clauses resulting from $f \leftrightarrow a \vee b$. Note that the subclause $\{b, c\}$ is no longer a common subclause afterwards and its frequency is now 0. Our algorithm updates the frequency of the common subclause after each elimination of a common subclause. To find all possible common subclauses and their frequencies effectively, we borrow the ideas from *frequent item set mining* (also called *association rule mining*) (Agrawal, Imieliński, and Swami 1993).

LZW Based CSE

The LZW based CSE employs the techniques of the LZW compression algorithm to rapidly replace *some* common subclauses in a given SAT problem. As a result, the order of the variables in each clause matters in identifying common subclauses. Each clause now is defined as a sequence of literals. A clause c is a *common subclause* of two clauses c_1 and c_2 if c is a subsequence of c_1 and c_2. For example, the (ordered) clauses $(a, b), (a, b, c), (a, c)$ have only one common subclause (a, b). Replacing (a, b) with a new variable d, we get $(d), (d, c), (a, c)$ and add $d \leftrightarrow a \vee b$ to form a new equivalent problem.

Like LZW, a dictionary is used to keep track of subclauses and codes assigned to subclauses. Unlike in LZW, a subclause is substituted only after being found common (i.e., appear in more than one clauses).

Experimental Results

Two benchmarks from SAT competitions were used:

- SAT13: SAT Competition 2013

Table 1: SAT Solving Performance Change by CSE

benchmark	CSE approach	timeout	instances solved		total solving time shorten by (%)	instances speedup		
			before CSE	after CSE		%	solving time shorten by	
							mean (%)	std (%)
3SAT	LZW	15m	60	60	8.07	54.24	11.39	24.59
		48h	188	187	−0.40	54.55	7.74	19.06
	Freq(2,2)	15m	60	57	5.29	41.04	19.62	28.08
		48h	188	188	0.38	51.87	12.06	21.59
SAT13	LZW	15m	163	188	6.68	47.55	47.40	30.38
		48h	507	507	1.69	51.81	39.44	34.03
	Freq(2,2)	15m	67	76	0.96	58.93	37.48	26.97
		48h	302	305	3.59	55.00	29.33	28.30
	Freq(3,2)	15m	43	45	21.15	60.61	43.73	20.47
		48h	228	225	2.38	57.78	27.70	26.32

- 3SAT: Problems that were explicitly labeled as 3SAT in SAT Competitions from 2007 to 2013 (to examine CSE on short clauses)

Two CSE approaches are compared:

1. LZW: LZW-based approach
2. Freq(s, f): Frequency-based approach with size threshold s and frequency threshold f. Only common subclauses containing more than s literals and appearing in more than f clauses will be replaced. 3SAT problems cannot be processed by Freq$(3, f)$ for any f.

We focus our experiments on instances containing common subclauses. They are considered *eliminable*. Table 2 lists number of **eliminable instances** found using different CSE approaches in all 2 benchmarks used. Note that frequency based CSE has less eliminable instances than the LZW based CSE because of resource limitation (the former consumes much more CPU time and memory than the latter).

Table 2: Number of eliminable instances

(raw #)	SAT (1030)	3SAT (496)	Total (1526)
LZW	931	496	1427
Freq(2,2)	674	490	1164
Freq(3,2)	471	N/A	471

Results on CSE's impact to SAT solving are given in Table 1. The column "total solving time shorten by (%)" means the percentage of solving speed increases on all instances in corresponding benchmark (column "benchmark") using a particular CSE approach (column "CSE approach") under a given time limit (column "time limit per instance"). Total solving time is shorten by a positive number indicates solving speed increases after CSE. Further analysis on accelerated instances are reported under banner "instances speedup".

With only one exception (LZW on 3SAT with 48hr time limit), total solving times are reduced on all benchmarks and all CSE approaches. Even for the exceptional case, more instances (54.55%) are solved faster after CSE. The speedup is particularly obvious for 15-minute time limit. For example, when time limit is 15 minutes, Freq(3,2) achieved 21.15% speedup on SAT13.

For each benchmark in each CSE approach, most instances (percentages given in % column under "instances speedup" banner of Table 1) are solved faster after CSE. There are two exceptions: 3SAT-Freq$(2, 2)$ 15-min and SAT13-LZW 15-min. The good news is that they both have relatively high overall speedup (5.29% and 6.68%, respectively).

We further look into the speed change happened on speedup instances. On SAT13, all three CSE approaches under either time limit can solve speedup instances by 27.70% to 47.40% faster. On 3SAT, these numbers range from 7.74% to 19.62%. Also, standard deviation of speed increase is moderate, ranging from 19.06% to 28.08%.

Overall speaking, through CSE, total solving time can be shorten, with more speedup instances than slowdown instances, and with stable and major speed increase on speedup instances.

Additional info of this work will be posted at https://sites.google.com/site/yanyaw00/aaai15

Acknowledgment

Chris Gutierrez and Yuanlin Zhang are partially supported by NSF grant IIS-1018031.

References

Agrawal, R.; Imieliński, T.; and Swami, A. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD'93, 207–216.

Katsirelos, G.; Sabharwal, A.; Samulowitz, H.; Simon, L.; et al. 2013. Resolution and parallelizability: Barriers to the efficient parallelization of sat solvers. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 481–488.