Boosting Minimal Unsatisfiable Core Extraction: Paper Addendum

Alexander Nadel

Intel Corporation,
P.O. Box 1659, Haifa 31015 Israel
alexander.nadel@intel.com

Abstract. This document is an addendum to [38]. Its main goal is to provide additional experimental data. In particular, we compare our algorithms to publicly available algorithms for single MUC extraction. We also provide further details about the performance of our algorithms. In addition, we correct a number of inaccuracies. This document should be read after [38].

1 Experimental Results

In this section, we provide additional experimental results.

5.1 Comparing the Resolution-based Method to the Selector-Variable-based Method for High-Level MUC Extraction

This section provides a more detailed analysis of the results presented in Tables II and III in [38]. Fig. 1 and Fig. 2 provide the data regarding the number of clauses and interesting constraints, respectively, in the instances we used.

Fig. 1. Clause size of the instances used for testing high-level MUC extraction algorithms.

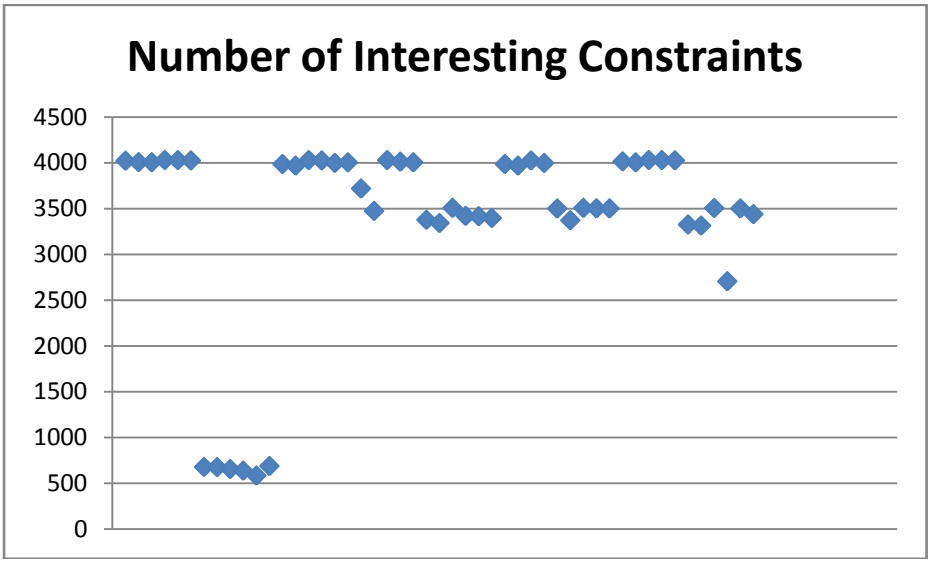

Fig. 2. The number of interesting constraints for the instances used for testing high-level MUC extraction algorithms.

Fig. 3 and Fig. 4 compare the best version of our resolution-based approach to high-level single MUC extraction 1MN vs. the selector variable-based approach SV. These figures compare run-times and MUC sizes, respectively. Note that the run-time summary and the average MUC size for these algorithms appear on Table III in [38]. The main observation is that the results are scattered, meaning that both approaches have their own strengths and weaknesses. More investigation is needed to understand the reasons for this phenomenon.

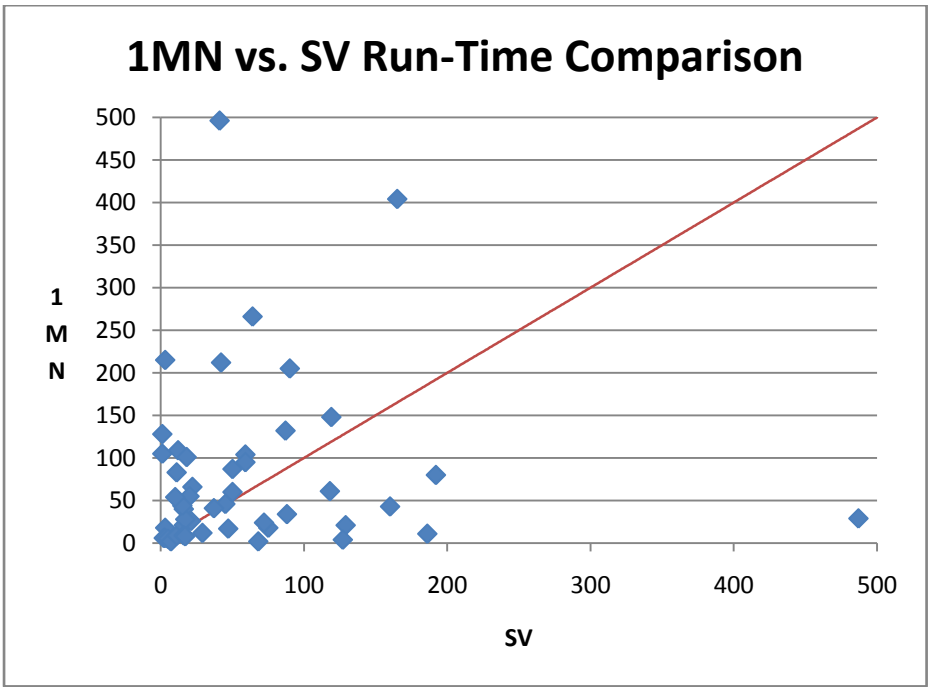

Fig. 3. Comparing the run-time of 1MN vs. SV algorithms for high-level MUC extraction.

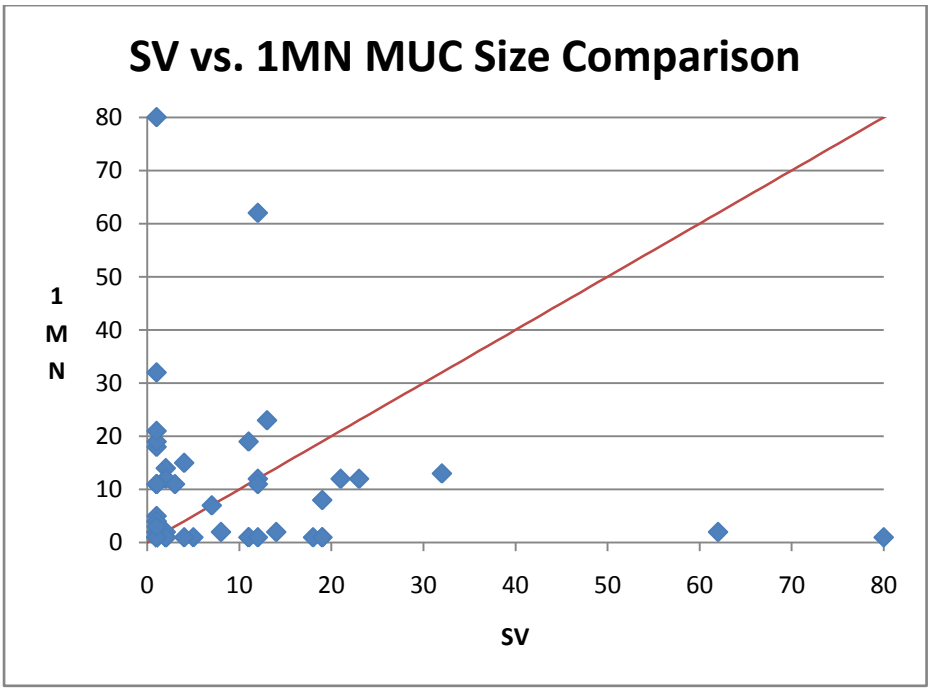

Fig. 4. Comparing the MUC size of 1MN vs. SV algorithms for high-level MUC extraction.

5.2 Comparing Publicly Available Tools for High-Level MUC Extraction

This section compares CAMUS [13], the only publicly available tool that is able to extract high-level MUCs, with our selector-variable-based algorithm SV. We configured CAMUS to extract a single MUC only and to use resolution-based preprocessing to obtain optimal performance for these experiments (our claim in [38] that CAMUS can only extract all of the MUCs is incorrect.). All the experiments with CAMUS (as well as AOMUS and MiniUnsat in Section 5.3) were carried out on Intel Xeon machines with 4Ghz CPU frequency and 32Gb of memory. These machines have the same CPU frequency as those used for benchmarking algorithms in [38], but they have eight times more memory. We use the results of SV that were collected in [38].

Consider Fig. 5. The main observation is that SV outperforms CAMUS by 1-2 orders of magnitude. SV solves each of the instances in less than 500 seconds, while CAMUS times out on many of the instances. As CAMUS is the only existing tool that is able to extract high-level MUCs, our approach to this problem significantly outperforms the state-of-the-art.

5.3 Comparing Publicly Available Tools for Clause-Level MUC Extraction

This section compares various algorithms for clause-level single MUC extraction. In particular, we compare our best resolution-based algorithm 1MN, our selector variable-based algorithm SV, our previous resolution-based approach CRR+RRP [6-8], CAMUS [13], AOMUS [39], and MiniUnsat [14]. We did not use AMUSE [4] and MUP [36] for the comparison, since these tools are outperformed by CRR+RRP [6-8]. Recall from [38] that CRR+RRP exactly corresponds to PDR.

The comparison is carried out on the same instances as in [38]. Consider Table 1. First note that AOMUS and MiniUnsat are outperformed by other approaches, hence we will concentrate on comparing the other algorithms. Consider the first 7 instances taken from the domain of microprocessor verification [38]. The instances from this family are the largest in terms of the number of clauses. Note that CAMUS cannot solve these instances, while 1MN is the best approach for them. We move on to the *barrel* instances [33]. The results for these instances are mixed. CAMUS performs the best for the first two instances. SV is the best approach for another instance, while PDR performs the best for another instance (1MN is only 5 seconds slower than PDR). CAMUS turned to be out the best MUC extractor for the *longmult* family [33]. This result may be related to the fact that this family has the fewest number of clauses of all three families we used. Additional study is required to understand these results better.

2 Conclusion

We have provided additional experimental results to complement the experimental results section of [38]. We have seen that our selector-variable-based approach outperforms the state-of-the-art for the problem of high-level MUC extraction by 1-2 orders of magnitude. Our resolution-based approach is faster for clause-level MUC extraction on large and difficult test-cases.

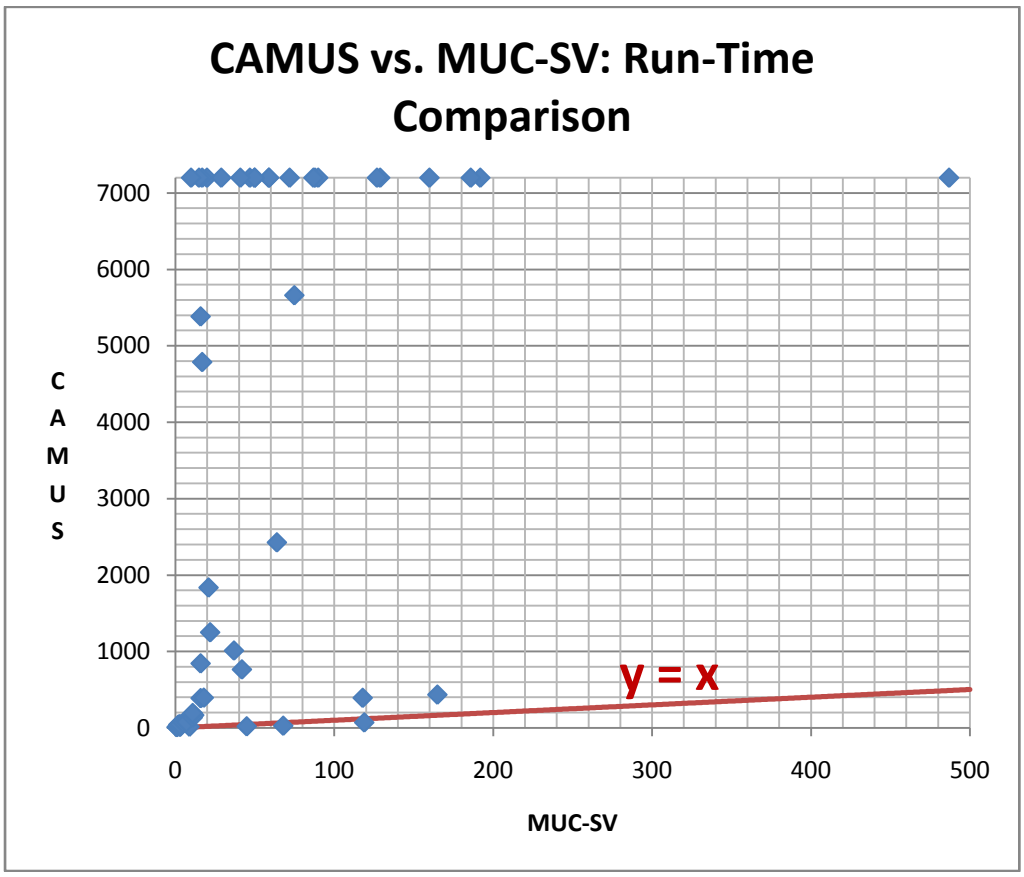

Fig. 5. Comparing CAMUS to our selector-variable-based approach to single high-level MUC extraction. The run-time in seconds of both approaches is shown. The time-out was 7200 seconds.

Table 1. Comparing different approaches to single clause-level MUC extraction. The run-time is provided in seconds. The time threshold was 2 hours. The best time is highlighted.

Instance	Clauses	1MN	SV	CRR+RRP	CAMUS	AOMUS	MiniUnsat
3pipe_k	*27405*	167	239	469	TO	ERR	TO
4pipe	*80213*	1417	2021	3791	TO	TO	TO
4pipe_1_ooo	*74554*	1528	4323	2928	TO	TO	TO
4pipe_2_ooo	*82207*	2383	4999	4566	TO	TO	TO
4pipe_3_ooo	*89473*	2560	5357	4465	TO	TO	TO
4pipe_4_ooo	*96480*	2432	6354	5865	TO	TO	TO
4pipe_k	*79489*	1426	3097	2938	TO	TO	TO
barrel5	*5383*	68	48	115	31	ERR	267
barrel6	*8931*	348	402	436	223	ERR	1391
barrel7	*13765*	849	700	1081	1443	ERR	5027
barrel8	*20083*	4115	5758	4110	5823	ERR	TO
longmult4	*6069*	14	78	12	5	540	227
longmult5	*7431*	143	642	100	21	ERR	538
longmult6	*8853*	968	5705	1760	79	TO	1149
longmult7	*10335*	5099	TO	TO	858	TO	2497

References[1]

[38] Alexander Nadel, "Boosting Minimal Unsatisfiable Core Extraction". Accepted to FMCAD'10.

[39] Éric Grégoire, Bertrand Mazure, Cédric Piette: Local-search Extraction of MUSes. Constraints 12(3): 325-344 (2007)

[1] The first 37 references appear in [38].