A Cardinality Solver: More Expressive Constraints for Free (Poster Presentation)

Mark H. Liffiton and Jordyn C. Maglalang

Illinois Wesleyan University, Bloomington IL 61701, USA
{mliffito,jmaglala}@iwu.edu
http://www.iwu.edu/~mliffito/

Despite the semantic simplicity of cardinality constraints, the CNF encodings typically used to solve them invariably turn one constraint into a large number of CNF clauses and/or auxiliary variables. This incurs a significant cost, both in space complexity and in runtime, that could be avoided by reasoning about cardinality constraints directly within a solver. Adding a single, *native* cardinality constraint instead of numerous clauses and/or auxiliary variables avoids any space overhead and simplifies the solver's procedures for reasoning about that constraint. Inspired by the simple observation that clauses are cardinality constraints themselves, and thus cardinality constraints generalize clauses, this work seeks to answer the question: How much of the research on developing efficient CNF SAT solvers can be applied to solving cardinality constraints?

Additional motivation came from our experience with a native implementation of cardinality constraints that was included in some early versions of the MiniSAT solver [3] as a simple, unoptimized example of the solver's ability to easily incorporate non-clausal Boolean constraints. That ability incurred unwanted overhead and was removed in later versions, and the native cardinality constraint implementation received little attention compared to the work done on CNF encodings. In addition to those early versions of MiniSAT, there have been other implementations of cardinality constraints that could be considered "native," but we are aware of none that integrate the constraints into a SAT solver by simply extending the existing clauses to incur little to no overhead.

For example, any Pseudo-Boolean (PB) solver or Satisfiability Modulo Theories (SMT) solver that handles linear integer arithmetic can solve cardinality constraints directly, as their constraints subsume both clauses and cardinality. Numerous PB solvers have been developed by extending a SAT solver, but little attention was paid to their performance on CNF. We are aware of only one experimental comparison between a PB solver and its corresponding SAT solver on CNF instances [2], comparing PBChaff with ZChaff, and the PB version was found to be consistently slower; the extension to more expressive constraints came at a noticeable cost. On the contrary, by restricting the solver to cardinality constraints and not general PB constraints, the implementation in this work retains those properties and efficiencies.

Asín, et al. [1] evaluated an "SMT-based approach" to cardinality constraints that solved them without encoding them to CNF. The "SMT" implementation was created by "coupling" two solving engines, which does not permit the tight

integration of cardinality into the SAT solver done in this work, and it did not perform well compared to CNF encodings. Marques-Silva and Lynce [4] explored modifications to a SAT solver that improved its efficiency when using a particular CNF encoding, but it still faced the inherent space complexity of such encodings and was limited to AtMost constraints with a bound of 1.

The aim of this work is to generalize a state-of-the-art SAT solver at negligible cost, producing a "cardinality solver" we call *MiniCARD*, and to exhibit the performance of MiniCARD compared to some of the best-performing CNF encodings of cardinality constraints. MiniCARD outperforms CNF encodings of cardinality constraints on all pure-cardinality instances tested, and instances with a mix of clauses and cardinality constraints exhibit mixed results indicating some effect beyond the performance of the constraints themselves. The modifications to the solver are minimal, and it retains its performance on pure CNF instances. Given the feasibility of achieving increased expressive power over CNF with minor, performance-neutral changes to a state-of-the-art CNF solver, it is well worth pursuing further research on cardinality solvers.

Several direction of future research are immediately suggested by this work. The first is to more completely evaluate the performance of MiniCARD relative to other means of solving cardinality constraints, especially SAT solvers with preprocessing. Investigations of specific instances are suggested as well, such as determining how different cardinality implementations affect applications like CAMUS and MSU4. And finally, cardinality constraints may be a better target than CNF for many types of problems and constraints for which CNF encodings have been developed, such as PB constraints. The greater expressive power of cardinality solvers with equivalent performance on clauses could enable encodings that are both simpler and more efficient than pure CNF encodings.

Acknowledgments

Thanks to Albert Oliveras for providing the MSU4 benchmarks and to Niklas Sörensson for helpful discussions and advice regarding MiniSAT.

References

1. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality networks: a theoretical and empirical study. Constraints 16, 195–221 (2011)
2. Dixon, H.E.: Automating Psuedo-Boolean Inference within a DPLL Framework. Ph.D. thesis, University of Oregon (2004)
3. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Proceedings of the 6^{th} International Conference on Theory and Applications of Satisfiability Testing (SAT-2003). LNCS, vol. 2919, pp. 502–518 (2003)
4. Marques-Silva, J., Lynce, I.: Towards robust CNF encodings of cardinality constraints. In: Principles and Practice of Constraint Programming (CP 2007). LNCS, vol. 4741, pp. 483–497 (2007)