DEPARTMENT OF COMPUTER SCIENCE, ROCHESTER INSTITUTE OF TECHNOLOGY

Performance Evaluation of $k - SAT$ Solvers

Applied to Graph Arrowing

Christopher Wood
Advisor: Professor Stanisław Radziszowski

March 30, 2013

1 Introduction

In his 1972 seminal paper entitled, "Reducibility Among Combinatorial Problems," Karp introduced a list of 21 NP-complete problems, including Boolean satisfiability, the maximum cut of a graph, and 0-1 integer programming [1]. The complexity of these problems was proven by deriving a polynomial-time reduction from CIRCUIT-SAT = $\{\langle C\rangle : C$ is a satisfiable Boolean combinational circuit $\}$, the first problem shown to be NP-complete by Cook in 1971 [2], starting the rush of complexity theory research.

The problem $3 - SAT$, or more formally, $3 - CNFSAT$, is a special case of satisfiability. It is a decision problem in which takes as input a 3-CNF Boolean formula and returns YES if the formula is satisfiable, and NO otherwise [3]. A 3-CNF formula, more formally known as a Boolean formula in 3-conjunctive normal form, is expressed as the Boolean AND of arbitrarily many clauses, where each clause is the Boolean OR of three literals, which is a Boolean variable or its negation. Such a Boolean formula is said to be satisfiable if and only if there exists an assignment of truth values to the variables such that substituting them into the literals of the formula will cause it to evaluate to true (or 1). Expressed as a formal language, we have that $3 - SAT = \{\langle\phi\rangle : \phi$ is satisfiable $\}$.

In 2002 Hans van Maaren of and John Franco initiated the public SAT competition in search of optimal performing SAT solvers judged by a variety of criteria and specializations, including their ability to demonstrate satisfiability and exhaustively prove unsatisfiability. In addition, since SAT is a problem that often arises in academia and the industry, each of the candidate solvers are rigorously tested with massive application-specific, crafted, and random Boolean

formulas as input. With three different solver specializations tested against three different types of inputs, and a first, second, and third place awarded to the candidates, a total of 27 possible trophies are awarded each year. In the most recent competition held in 2011, solvers were tested using CPU time and world-clock time as a basis for their results, thus expanding the trophy space to 54 slots. The next competition is slated to take place in 2013.

The SAT problem is particularly interesting when applied to graph arrowing. It can be shown how to reduce the question $G \overset{?}{\nrightarrow} (3,3)^e$ to an equivalent $3-CNF$ formula ϕ_G such that $G \nrightarrow (3,3)^e \Leftrightarrow \phi_G$. Intuitively, this is a very promising technique for determining if $G \rightarrow (3,3)^e$ for K_4-free graphs G.

The immediate application of this technique is to attack the upper bound of the Folkman number $F_e(3,3;4)$. In particular, to lower this bound, we will need to find a graph G on n vertices where $G \rightarrow (3,3)^e$. It has been conjectured that $G_{127} = G(127,3) = (\mathbb{Z}_{127}, E = \{(x,y)|x-y = \alpha^3 \mod 127\})$ is a prime candidate for witnessing an upper bound of 127 because of its denseness and large number of triangles. To determine whether G_{127} is indeed a witness we will decompose the problem of arrowing into problems on subgraphs H that witness $H \nrightarrow (3,3)^e$, and then carefully extend H to encompass all of G. For each subgraph H will generate a corresponding 3-CNF formula ϕ_G by mapping the edges in $E(H)$ to variables in $\phi_H \in$ 3-SAT, and for edge adding the following clauses to ϕ_H:

$$(x+y+z) \wedge (\bar{x}+\bar{y}+\bar{z})$$

If H can be extended to encompass all of G and ϕ_G is shown to be unsatisfiable, then $G \rightarrow (3,3)^e$, and so $F_e(3,3;4) = 127$.

Unfortunately, while this approach seems simple upfront, the complexity of the formulas ϕ_H with $n \approx 85$ has proven to be very difficult for modern SAT solvers to handle. In this study, we will attempt to determine the structure of these formulas that makes them so difficult to solve. We will also present a comprehensive performance comparison for the popular SAT solvers entered into the SAT competition, including Minisat, zChaff, glucose, ppfolio //, ppfolio seq, contrasat hack, and 3S.

2 $k-SAT$ Algorithms

The algorithms used in state-of-the-art SAT solvers tend to be based on the famous DPLL backtracking algorithm proposed by David, Putnam, Logemann, and Loveland in [6]. The DPLL algorithm works by recursively splitting a CNF formula into smaller formulas by assigning truth values to individual variables and simplifying the resulting formula at each level of iteration. Simplification works by removing all clauses in ϕ that become true under the new assignment and all literals which are false. This process continues recursively until the formula is deemed satisfiable or a conflict emerges. Satisfiability occurs when all variables have been assigned truth values and the resulting formula has non-empty clauses remaining, or when all clauses can be evaluted to true. If a conflict arises, the algorithm backtracks to the

Algorithm 1 DPLL Algorithm

Require: ϕ
Ensure: $true$ or $false$
1: TODO

Table 1: Performance of Minisat solver by removing N_R vertices from G_{127}

N_R	CPU Time	Restarts	Conflicts	Decisions	Propagations	Conflict Literals
46	1228.97s	28668	18968226	41661327	1999748077	930184526
45	4362.29s	66558	52952393	106164381	5779728314	2755169058
44	5576.51s	81915	62420966	126120884	6748259547	3254804365
43	21681.5s	245755	199022470	384078647	21687371442	10581492993

last assigned truth value, attempts to substitute the negated truth value, and then proceeds as normal. A formula ϕ is deemed unsatisfiabile if this algorithm exhaustively checks all possible truth value assignments without yielding a satisfiable formula.

The DPLL algorithm further enhances the backtracking search through the use of two additional procedures, unit propagation and pure literal elimination. Unit propagation works by assigning an appropriate truth value to all unit clauses (i.e. all clauses with only a single literal) and then propagating this truth value to the rest of the formula. For example, unit propagation of the formula $\phi = [(x_1) \wedge (x_1 \vee x_2 \vee x_3)]$ yields $\phi = [(x_2 \vee x_3)]$ by assigning a value of true to x_1. Pure literal elimination works by finding variables with have uniform polarity (i.e. always positive or negative). For all such literals, there always a exists a truth assignment such that the containing clauses will evaluate to true. Therefore, such clauses are delete from ϕ. For example, given the formula $\phi = [(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_2 \vee \neg x_4)]$, performing pure literal elimination yields $\phi = [(x_1 \vee x_2 \vee \neg x_4)]$ because we can assign $x_3 = false$ and eliminate the first two clauses. These two procedures are shown in the pseudocode description shown in Algorithm 1.

TODO: most solvers use a Minisat variation - articulate that here for completeness

3 Selected $k-SAT$ Solvers

TODO: discuss Minisat, zChaff, glucose, ppfolio //, ppfolio seq, contrasat hack, and 3S

4 Performance

The formula ϕ_G for G_{127} experiences a peculiar phase transition when $m \approx 41$. That is,

References

[1] Richard Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations, (RE Miller and JM Thatcher, eds.)* (1972), 85-103.

[2] Stephen A. Cook. The Complexity of Theorem-Proving Procedures. *In Proceedings of the third annual ACM symposium on Theory of computing (STOC '71)*. ACM, New York, NY, USA (1971), 151-158. DOI=10.1145/800157.805047. `http://doi.acm.org/10.1145/800157.805047`

[3] Gormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. MIT Press 44 (1990), 97-138.

[4] Niklas Sörensson and Niklas Eèn. Minisat v1.13 - A SAT Solver with Conflict-Clause Minimization. *SAT 2005* (2005) 53.

[5] Yogesh Mahajan, Zhaohui Fu, and Sharad Malik. Zchaff2004: An Efficient SAT Solver. *Theory and Applications of Satisfiability Testing*. Springer Berlin/Heidelberg (2005).

[6] M. Davis, G. Logemann, and D. W. Loveland, A machine program for theorem-proving, *Communications of the ACM* **5(7)** (1962), 394-397.