Artificial Intelligence 173 (2009) 240–257

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

# A comparative runtime analysis of heuristic algorithms for satisfiability problems

Yuren Zhou [a,c,*], Jun He [b], Qing Nie [c]

[a] School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China
[b] Department of Computer Science, University of Wales, Aberystwyth, Ceredigion, SY23 3DB, UK
[c] Department of Mathematics, University of California, Irvine, CA 92697-3875, USA

ARTICLE INFO

*Article history:*
Received 18 March 2008
Received in revised form 23 July 2008
Accepted 1 November 2008
Available online 7 November 2008

*Keywords:*
Boolean satisfiability
Heuristic algorithms
Random walk
$(1+1)$ EA
Hybrid algorithm
Expected first hitting time
Runtime analysis

ABSTRACT

The satisfiability problem is a basic core NP-complete problem. In recent years, a lot of heuristic algorithms have been developed to solve this problem, and many experiments have evaluated and compared the performance of different heuristic algorithms. However, rigorous theoretical analysis and comparison are rare. This paper analyzes and compares the expected runtime of three basic heuristic algorithms: RandomWalk, $(1+1)$ EA, and hybrid algorithm. The runtime analysis of these heuristic algorithms on two 2-SAT instances shows that the expected runtime of these heuristic algorithms can be exponential time or polynomial time. Furthermore, these heuristic algorithms have their own advantages and disadvantages in solving different SAT instances. It also demonstrates that the expected runtime upper bound of RandomWalk on arbitrary $k$-SAT ($k \geqslant 3$) is $O((k-1)^n)$, and presents a $k$-SAT instance that has $\Theta((k-1)^n)$ expected runtime bound.


## 1. Introduction

The satisfiability problem (SAT) of a propositional formula plays a central role in computer science and artificial intelligence. It is the first proposed NP-complete problem [5,21] and one of the basic core NP-complete problems [10]. In addition to its theoretical importance, the SAT problem is also directly applied in VLSI formal verification, software automation, and so on.

Researchers have been trying to look for an effective algorithm for the SAT problem. Since the SAT problem is an NP-complete problem in nature, a polynomial algorithm is not currently available to solve it, although we cannot prove that such an algorithm does not exist. In fact, a basic conjecture of modern computer science and mathematics is that no polynomial algorithm exists for NP-complete problems. At present, the main methods for solving the SAT problems are complete algorithms [3,6,34] and incomplete algorithms [7,12,13,15,20,25,27,29,31,32]. There are several very successful complete algorithms (e.g., SATO [34]). A complete algorithm often explores the whole search space and can always determine whether a given propositional formula is satisfiable or not; however, its time complexity is usually exponential. An incomplete algorithm does not carry out a complete search on the search space; instead, it often explores some part of the search space using heuristic information within a limited time; however it does not give the correct answer with certainty.

Since the 1990s, the use of incomplete algorithm for solving the SAT problem has grown quickly. The basic incomplete heuristic methods are RandomWalk algorithm [25], GSAT algorithm [13,31], WalkSat algorithm [32], UnitWalk [15],

* Corresponding author at: School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China.
*E-mail addresses:* yrzhou@scut.edu.cn (Y. Zhou), jun.he@ieee.org (J. He), qnie@math.uci.edu (Q. Nie).

0004-3702/$ – see front matter 
doi:10.1016/j.artint.2008.11.002

population-search-based evolutionary algorithms [7,12,20] and so on. In recent years, some powerful concepts and techniques of statistical physics have been applied to the SAT problem. One of these incomplete algorithms, known as "survey propagation" [4,22], which is based on statistical physics methods, shows good performance on some difficult randomly generated SAT instances. It is well known that one of the earliest applications of statistical physics in the optimization problem is the simulated annealing algorithm [19]. WalkSat [32] used a probability selection mechanism similar to that of the simulated annealing algorithm.

For some heuristic algorithms for the SAT problem, theoretical results about computational complexities have been obtained to some extent. Papadimitiou [25] was the first to prove that the average time upper bound of RandomWalk for 2-SAT is $O(n^2)$. Schöning [29] presented a restarting local-search algorithm to show that, for any satisfiable $k$-CNF formula with $n$ variables, the algorithm has to repeat $O((2(1-\frac{1}{k}))^n)$ times, on average, to find a satisfying assignment. Specially if $k=3$, the average time is $O(1.334^n)$ (the upper bound of an exhaustive search is $O(2^n)$). There have been several improvements on the upper bound by hybrid algorithms based on randomized algorithms by Paturi et al. [27] and Schöning [29], e.g. $O(1.324^n)$ [18] and $O(1.322^n)$ [28]. Alekhnovich et al. [2] proved that, when the clause density is less than 1.63, the average time complexity of RandomWalk for 3-SAT is linear.

Since there are many incomplete heuristic algorithms for SAT problems, comparing and understanding the working principals of these heuristic algorithms is useful. The first thing we have to accept is that no one algorithm beats all other algorithms on all problems. There have been many numerical experiments that compared various heuristic algorithms on SAT problems, but theoretical study has been rare. This paper analyzes and compares the expected running time of three basic heuristic algorithms: RandomWalk, $(1+1)$ EA, and hybrid algorithm. We use absorbing Markov chains to model search processes of these heuristic algorithms, and use explicit expressions of the first hitting time of a Markov chain to analyze and estimate their expected runtime. Through runtime analysis of three SAT instances, we show that the expected runtime of these heuristic algorithms can be exponential or polynomial. We also find that these heuristic algorithms have their own comparative advantage under different circumstances.

The rest of this paper is organized as follows. Section 2 introduces the concepts of the SAT problem, some heuristic algorithms for the SAT problem, and the first hitting time of an absorbing Markov chain. Section 3 discusses the worst-case bound and the worst-case example on RandomWalk. Section 4 analyzes and compares the expected runtime bounds of three heuristic algorithms on two 2-SAT instances. Section 5 presents our conclusions and suggestions for further research.

## 2. Heuristic algorithms for satisfiability and the first hitting time of the Markov chain

### 2.1. The SAT problem

We begin by stating some definitions and notations that will be used throughout the paper.

In Boolean logic, a literal is a variable or its negation, and a clause is a disjunction of literals. The formula $f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is in $k$ conjunctive normal form ($k$-CNF) if it is a conjunction of clauses with each clause as a disjunction of at most $k$ literals. We view a CNF Boolean formula as both a Boolean function and a set of clauses. Satisfiability is the problem of determining whether the variables of a given Boolean formula can be assigned truth values in such a way as to make the formula evaluate to true.

SAT is originally stated as a decision problem. In this paper we consider the more general MaxSAT, so, our goal is to look for an assignment that satisfies the maximum number of clauses.

Evolutionary algorithms (EAs) are the heuristic algorithms that have been applied to SAT and to many other NP-complete problems. EAs usually use a fitness value to guide the search process. In the MaxSAT formulation, the fitness value is defined as the number of satisfied clauses, i.e.

$$fit(x) = c_1(x) + c_2(x) + \cdots + c_m(x) \tag{1}$$

where $c_i(x)$ $(1 \leqslant i \leqslant m)$ represents the true value of the $i$th clause. This fitness function is used in most EAs for SAT problems.

Throughout this paper, for $x = (x_1 \cdots x_n)$, $y = (y_1 \cdots y_n) \in \{0,1\}^n$, we denote by $H(x, y)$ the Hamming distance between two points $x$ and $y$, i.e. $H(x, y) = \sum_{i=1}^n |x_i - y_i|$. We also denote $|x| = x_1 + \cdots + x_n$, and let $S_i = \{x \mid x \in S = \{0,1\}^n,\ |x| = i\}$ $(i = 0, 1, \ldots, n)$ be a partition of search space $S = \{0,1\}^n$.

### 2.2. Heuristic algorithms for the SAT problem

RandomWalk, first introduced by Papadimitiou [25], is one of the most basic incomplete algorithms, and many other heuristics have been developed based on the improvement of this algorithm, e.g. the Walk-SAT [32], combines RandomWalk with a greed bias towards assignments that satisfy more clauses. RandomWalk algorithm first randomly selects a clause that is not satisfied with the CNF, then randomly selects a flip in the clause (see Algorithm 1).