Theoretical Computer Science 657 (2017) 28–37

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Two approximate algorithms for model counting

Jinyan Wang [a,b,*], Minghao Yin [c], Jingli Wu [a,b]

[a] Guangxi Key Lab of Multi-source Information Mining & Security, Guangxi Normal University, Guilin, China
[b] College of Computer Science and Information Technology, Guangxi Normal University, Guilin, China
[c] School of Computer Science and Information Technology, Northeast Normal University, Changchun, China

ARTICLE INFO

Article history:
Received 15 October 2015
Received in revised form 25 April 2016
Accepted 27 April 2016
Available online 6 June 2016

Keywords:
Propositional satisfiability
Model counting
Resolution principle
Extension rule

ABSTRACT

Model counting is the problem of computing the number of models or satisfying assignments for a given propositional formula, and is #P-complete. Owing to its inherent complexity, approximate model counting is an alternative in practice. Model counting using the extension rule is an exact method, and is considered as an alternative to resolution-based methods for model counting. Based on the exact method, we propose two approximate model counting algorithms, and prove the time complexity of the algorithms. Experimental results show that they have good performance in the accuracy and efficiency.

1. Introduction

In recent years, there are tremendous improvements in the field of propositional satisfiability (SAT), which is the problem of determining whether there exists a truth assignment for a given propositional formula to make the formula be *True*. Many hard combinatorial problems in artificial intelligence and computer science, such as planning problems, have been compiled into SAT instances, and solved effectively by SAT solvers [1–4]. Model counting (#SAT) is the problem of counting the number of models or satisfying assignments for a propositional formula, as an important extension of satisfiability testing, and is the canonical #P-complete problem, which is widely believed to be significantly harder than the NP-complete SAT problem [5,6]. Efficient algorithms for the problem have been applied in many areas, such as performing inference in Bayesian networks [7,8], probabilistic planning [9] and diagnosis [10].

Resolution principle is the rule of inference at the basis of most procedures for both SAT and #SAT [5]. In [11], Birnbaum and Lozinskii directly extended Davis–Putnam (DP) procedure [12] to solve #SAT problems, and proposed a model counter CDP. Based on CDP, Bayardo and Pehoushek introduced the idea of connected component analysis to enhance Relsat's model counting ability [13]. Furthermore, the introduction of component caching and clause learning accelerated greatly the model counting procedure [14]. Since the space requirement was an important concern in implementing component caching, sharpSAT solver employed advanced component caching mechanism to make components be stored more succinctly [15]. In addition, Bacchus et al. [16] utilized backtracking search to solve #SAT problem. Lagniez and Marquis [17] concerned preprocessing techniques for model counting. The running time and memory usage of these algorithms often increase exponentially with problem size. Consequently, they are limited to relatively small formulas. In 2005, Wei and Selman presented an approximate model counting algorithm ApproxCount [18] based on SampleSat [19], which sampled from the set of solutions of a propositional formula near-uniformly. Following the scheme outlined by Jerrum et al. [20],

* Corresponding author at: College of Computer Science and Information Technology, Guangxi Normal University, Guilin, China.
E-mail address: wangjy612@gxnu.edu.cn (J. Wang).

http://dx.doi.org/10.1016/j.tcs.2016.04.047
0304-3975/

ApproxCount used exact model counting methods like Relsat [13] or Cachet [14] to count models of the residual formula after some variables had been set by SampleSat. There are other model counting techniques based on sample method, such as SampleCount [21], SampleMinisat [22]. Kroc et al. [23] utilized belief propagation and MiniSat to design approximate model counting algorithm.

In [24,25], Lin et al. used the extension rule to solve SAT problem. The key idea is to use inclusion–exclusion principle to solve the problem. For a propositional formula, if there are maximum terms, which are not generated from the formula by using the extension rule, then the formula is satisfiable. Otherwise, it is unsatisfiable. They analyzed the relationship between a maximum term and a truth assignment, and proposed a model counting algorithm using the extension rule (MCER) [26,27]. Actually, the number of different maximum terms generated from a formula is equal to the number of unsatisfying assignments. Therefore, it can obtain the number of satisfying assignments or models. They compared the method with resolution-based methods. The more pairs of clauses with complementary literals are, the more efficient the method is, and the less efficient resolution-based methods are. So the method is considered as an alternative to resolution-based methods for model counting. Furthermore, Bennett and Sankaranarayanan [28] presented a pruning technique to model counting using the inclusion–exclusion principle. Also, Linial, Nisan and Kahn [29,30] considered approximate inclusion–exclusion, when intersection sizes are known for only some of the subfamilies, or when these quantities are given to within some error, or both.

In this paper, we use the extension rule to present two approximate model counting algorithms: ULBApprox and SampleApprox. The idea of ULBApprox is as follows. In MCER, we use the inclusion–exclusion principle to count the number of different maximum terms generated from a formula, i.e., the number of unsatisfying assignments. It is known that the value of the first odd (or even) sum terms is an upper (or lower) bound in the inclusion–exclusion principle. However, it is impossible that any value of the first odd (or even) sum terms can be viewed as an approximation of the number of unsatisfying assignments, because the quality of estimate of some of them is very poor. Therefore, for an upper (or lower) bound, we firstly analyze the condition under which the bound can be viewed as an approximate value, then we count an approximate number of models. SampleApprox is an approximate algorithm which combines the extension rule with SampleSat algorithm, which uses SampleSat algorithm to get an approximate value of the total number of models divided by the number of models of the restricted subspace, and a sub-algorithm SubMCER to count the number of models of a restricted subspace. Experimental results indicate the two approximate algorithms have good performance in the accuracy and efficiency.

The paper is organized as follows. We review the extension rule and describe the model counting method based on the extension rule in the next section. In section 3, we give two approximate model counting algorithms by using the extension rule. Experimental results are given in section 4. In the last section, we discuss this paper and give further work.

2. Model counting using the extension rule

A propositional or Boolean formula F is a logic expression over a set V of Boolean variables. A truth assignment to the set V is a map $\sigma : V \mapsto \{True, False\}$, identified with 1 and 0, respectively. For a propositional formula F, a satisfying (or unsatisfying) assignment is a truth assignment σ such that F evaluates to 1 (or 0) under σ. A propositional formula in conjunctive normal form (CNF) is a conjunction of clauses, denoted also by a set of clauses, where a clause is a disjunction of literals, and a literal is either a variable a or its negation $\neg a$. We specify the notations that will be used in the rest of this paper. Σ denotes a set of clauses or a propositional formula in CNF, C denotes a clause, V denotes the set of all variables appearing in Σ, and $M(\Sigma)$ denotes the number of models of Σ.

2.1. Extension rule

We review the extension rule in brief. The readers are referred to [24] for more details. Given a clause C and a set V of variables, $D = \{C \vee a, C \vee \neg a | a \in V$ and a does not appear in $C\}$. The operation proceeding from C to D is called extension rule on C, and D is the result of the extension rule. A clause C is logically equivalent to the result of the extension rule D. So the extension rule is a sound inference rule.

A clause is a maximum term on a set V if it contains all variables in V in either positive form or negative form. For example, given a set $V = \{a, b, c\}$ of variables, $a \vee b \vee c$ and $a \vee \neg b \vee c$ are maximum terms on V, but $a \vee b$, $\neg a \vee \neg c$ are not. Given a set Σ of clauses with its set $V(|V| = v)$ of variables, if clauses in Σ are all maximum terms on V, Σ is unsatisfiable if it contains 2^v clauses; otherwise, Σ is satisfiable. Therefore, we can decide the satisfiability of a set of maximum terms.

2.2. Model counting using the extension rule

According to the definition of maximum terms, we can count the number of models of a set of maximum terms. Given a set Σ of clauses with its set $V(|V| = v)$ of variables, if the clauses in Σ are all maximum terms on V, the number of models of Σ is $2^v - S$ if Σ contains S distinct clauses, where $S \leq 2^v$.

For example, given a set $\Sigma = \{a \vee b, a \vee \neg b, \neg a \vee b\}$ of clauses with its set $V = \{a, b\}$ of variables, the clause $\neg a \vee \neg b$ does not appear in Σ. It is clear the assignment, which a is assigned 1 and b is assigned 1, is a model of Σ. Actually, the number of maximum terms is equal to the number of unsatisfying assignments for a set of maximum terms. Therefore,