AN EXPOSNENTIAL LOWER BOUND FOR THE PURE LITERAL RULE

Khaled M. BUGRARA
College of Computer Science, Northeastern University, Boston, MA 02115, U.S.A.

Paul W. PURDOM
Department of Computer Science, Indiana University, Bloomington, IN 47405, U.S.A.

Communicated by A.V. Aho
Received 24 November 1986
Revised 31 August 1987

Keyword: Davis-Putnam Procedure

1. Introduction

A pure literal is a literal in a logic formula (usually in Conjunctive Normal Form) that occurs only positively or only negatively. The Davis-Putnam Procedure [1] was developed to find one solution to a logic formula, and it contains several techniques for speeding up the typical solution time. One of these techniques is the pure literal rule: a variable that occurs only positively or only negatively (a pure literal) needs to have only one value considered, the value that makes all of its clauses true.

To facilitate average time analysis, Goldberg [2] developed a simplified algorithm which consisted of just splitting (trying each value for a variable, simplifying the resulting subproblems, and considering each subproblem recursively) and the pure literal rule. The running time for this Pure Literal Rule Algorithm gives an upper bound for the running time of the full Davis-Putnam Procedure. The Pure Literal Rule Algorithm takes polynomial average time for some random sets of problems [2,3]. A detailed upper bound analysis [7] showed that there was an extensive region where the average time is polynomial.

This paper has the first lower bound analysis for this algorithm. We show that there is also an extensive region where the average time for this algorithm is exponential. In many cases, the current lower bound is much lower than the best upper bound [7]. (We are writing up an improved analysis which shows that true behavior of the algorithm is close to the upper bound in [7] when the average clause length is not too large.)

2. Random model

We have \( v \) variables, \( t \) clauses, and with probability \( p \) a literal is in a clause. A random clause is formed by including each literal with probability \( p \). A predicate consists of \( t \) independently selected random clauses. Since there are \( 2v \) literals altogether, and since each literal is included with probability \( p \), the expected clause size is equal to \( 2vp \). One should note that the expected clause size is linear in \( v \) whenever \( p \) is fixed; the expected clause size is fixed whenever \( p \) varies inversely with \( v \), i.e., \( p = O(v^{-1}) \).