MS Project Proposal, Horn Formula Minimization

Tom Y. Chang
(tyc0429@cs.rit.edu)

June 12, 2003
Contents

1 Abstract 3
2 Introduction 4
3 Background 5
4 Proposal 8
5 Tentative schedule 9
6 References 10
1 Abstract

Intractable problems are fascinating subjects for study and research in the field of computer science. The particular intractable problem considered in this project is that of Horn formula minimization, that is to find a minimal representation of a given Horn formula. Horn formulas are an important subclass of Boolean formulas given its crucial role in artificial intelligence. Moreover, Horn formulas exhibit interesting and useful computational properties, making them a good subclass for Boolean formula problems research. What is proposed in this project is to study and investigate existing techniques and proofs in Horn minimization, and to explore and experiment with new methods and approaches.
2 Introduction

The minimization problem for general Boolean formulas is a computationally hard problem. In fact, this problem in Garey and Johnson’s standard formalized version [Garey and Johnson 1998]: “Given a Boolean formula $\phi$ and a nonnegative integer $k$, is it true that there exists a Boolean formula of size at most $k$ that is equivalent to $\phi$,” has a time complexity lower bound of coNP-hard, and an upper bound of $\Sigma^p_2$ [Meyer and Stockmeyer 1972]. These complexity bounds are straightforward, and can be easily shown by the following. Since the Satisfiability problem for general Boolean formulas is NP-complete, this makes tautology a coNP-complete problem. Tautology can be translated into an instance of the Boolean formula minimization problem, thereby establishing the coNP-hard lower bound of the minimization problem. The upper bound of $\Sigma^p_2$ comes from the fact that the problem can be solved by a NP machine with a NP oracle. This machine would non-deterministically guess a formula of size $\leq k$, and ask the non-deterministic oracle to verify that it is equivalent to the original formula. Recently, the lower bound for the minimization problem of general Boolean formulas has been raised to many-one hard for parallel access to NP [Hemaspaandra and Wechsung 2002].

The minimization problem for different subclasses of Boolean formulas has been studied. For instance, it was found that the minimization problem for Boolean formulas in DNF is in $\Sigma^p_2$-complete [Umans 1998]. On the other hand, because the Satisfiability problem for Horn formulas is solvable in polynomial time [Dowling and Gallier 1984], the minimization problem for Horn formulas is in NP. Furthermore, Horn Minimization was proven to be in NP-complete via a reduction from the problems of Set Covering in [Hammer and Kogan 1993] and Hamiltonian Path in [Boros and Čepak 1994]. In the latter proof, a stronger result was obtained. It was shown that even for 3 Horn formulas (Horn formulas in which every clause has exactly 3 literals), the minimization problem is NP-complete.

As with all NP-complete problems, no efficient polynomial time algorithm is known that solves the problem of Horn formula minimization. There exists a quadratic time approximation algorithm which eliminates duplicate terms in a Horn formula with $n$ propositional variables, to achieve an equivalent formula of at most $n - 1$ times of the length of the minimum [Hammer and Kogan 1993]. It was found that for two subclasses of Horn formulas, acyclic and quasi-acyclic, the problem of minimization is solvable in polynomial time.
[Hammer and Kogan 1995]. More recently, a linear time algorithm has been presented which iteratively decomposes any given Horn formula, each time into two parts, and minimizes one of the two components, resulting in a shorter equivalent formula at the end [Boros, Čepek and Kogan 1997].

An interesting property of Horn formulas is that there exist minimized equivalent formulas that remain Horn [Hammer and Kogan 1993]. This makes minimization of Horn formulas useful and applicable in areas where the use of Horn formulas is necessary. For instance, in artificial intelligence, finding the minimal representation of a Horn formula is equivalent to compacting the size of a particular representation of knowledge without altering it. This results in reduced complexity in the reasoning of the Horn knowledge database, thus speeding up query handling.

This project will be geared towards theoretical research, with the hopes of further advances in approximation techniques for the Horn minimization problem.

3 Background

A Boolean function \( f \) of \( n \) propositional variables \( x_1, \ldots, x_n \) is a mapping from \( \{0, 1\}^n \) to \( \{0, 1\} \). Variables and their negations are called positive and negative literals, respectively. An assignment to the variables is called a truth assignment, and each truth assignment to the variables of a Boolean function yields either a True (1) or a False (0) value.

Every Boolean function \( f \) can be expressed as a Boolean formula \( F \) over the variables using conjunction (and), disjunction (or) and negation (not). Conjunctions and disjunctions of literals are formed using the \( \land \) and the \( \lor \) operators respectively. If each variable appears in a conjunction or a disjunction at most once, then the conjunction or disjunction is called a term or a clause. Boolean formulas in conjunctive normal form (CNF) are conjunctions of disjunctive clauses. Likewise, Boolean formulas in disjunctive normal form (DNF) are disjunctions of conjunctive clauses.

A disjunctive clause is called Horn if it contains at most one positive literal, and it is called pure Horn or definite Horn if it contains exactly one positive literal. A CNF formula is called Horn if every one of its clauses is Horn, and is called pure or definite Horn if every one of its clauses is pure Horn. Likewise, a conjunctive clause is called Horn if it contains at most one negative literal, and it is called pure Horn or definite Horn if it contains
exactly one negative literal. A DNF formula is called Horn if every one of its clauses is Horn, and is called pure or definite Horn if every one of its clauses is pure Horn.

The Horn Minimization problem has been stated by Boros and Čepek [1994] for Horn DNF, and by Hammer and Kogan [1993] and Boros, Čepek and Kogan [1997] for Horn CNF. It should be noted that the minimization problem for Horn formulas is equivalent among CNF and DNF representations, as is the case for minimization of general Boolean formulas. In this project, the CNF representation will be chosen. The Horn Minimization problem is: given a Horn CNF formula, find a Horn CNF formula representing the same function consisting of a minimum number of clauses. Horn Minimization can also be stated as a decision problem: given a Horn CNF formula \( F \) and a positive integer \( k \), is there a CNF formula equivalent to \( F \) which contains at most \( k \) clauses? Or simply, given a Horn formula, is it minimal? In this project, minimization from Horn CNF formulas to general formulas (those not of Horn CNF form) will also be looked into. Moreover, the Horn Minimization problem with different definitions of size will be considered.

The first algorithm in approximating Horn minimization by Hammer and Kogan [1992] transforms a Horn CNF \( F \) into an equivalent Horn CNF that is irredundant and prime, i.e. there are no repeated clauses, and dropping any literal from any clause results in a formula representing a different Boolean function. It utilizes a linear time implication test to check whether a given clause is an implicate of \( F \) by assigning variables occurring in the clause truth values (1 if occurred as a positive literal, and 0 if occurred as a negative literal), then checking the satisfiability of the resulting formula. The complexity of this implication check results from the complexity of the linear time Horn satisfiability testing algorithm.

By using the implication check, the algorithm first drops redundant literals from each clause of the input Horn CNF, resulting in a prime Horn CNF. Then it drops redundant clauses from this prime formula to make it irredundant. In [Hammer and Kogan 1993], it was proven that given a Horn formula with \( n \) variables, the number of clauses in the resulting equivalent irredundant and prime Horn formula is at most \( n - 1 \) times the minimum possible number of clauses. This obviously is far from optimal, thus this naive quadratic algorithm is only a good starting point.

The second approximation algorithm by Boros, Čepek, and Kogan [1997] takes a given Horn CNF and runs a series of decompositions of it to derive at
a shorter equivalent Horn CNF. The paper states “Each decomposition step splits the ‘current’ CNF into two non-empty parts with the property that the minimization of the original CNF is equivalent to the minimization of one of the parts. The iterative decomposition process is based on ‘switching’ of propositional variables and always produces the same result regardless of the order in which the propositional variables are switched.” The paper does not draw any conclusions about the resulting formulas in regard to their relationship to the minimal equivalent representations.
4 Proposal

I propose to do the following:

1. Conduct an extensive literature search on Horn minimization (MEE and Minimal) and related problems, and write an overview on the findings. I will also report on the applications of Horn formulas.

2. Study, implement and experiment with the current approximation algorithms and document the results. Methods to generate random problem sets will be researched and implemented as well. Furthermore, I will analyze the running time and effectiveness of the decomposition algorithm.

3. Experiment with heuristic techniques such as genetic algorithms, tabu search and simulated annealing towards alternative solutions.

4. Identify fragments in the problem domain that are in P.

5. Attempt to improve upon existing approximation algorithms.

6. Attempt to devise alternative proofs for some of the complexity hardness results.
5 Tentative schedule

- June 6, Friday: second draft of proposal
- June 10, Tuesday: presentation in MS project seminar
- June 13, Friday: final draft of proposal and submission to committee
- June 20, Friday: complete literature search and gathering
- July 4, Friday: complete a first draft of the overview section
- July 11, Friday: complete implementation of existing algorithms
- July 18, Friday: complete implementation of heuristic approaches
- July 25, Friday: complete a first draft of the experiment documentation section
- August 1, Friday: second draft of the project write-up submitted for review
- August 8, Friday: complete sections documenting attempts at the proof
- August 15, Friday: complete sections documenting attempts at improving the algorithms
- August 22, Friday: complete a third draft of the project write-up submitted for review
- August 29, Friday: project defense
6 References


