The Parallel Theorem Proving Algorithm Based on Semi-Extension Rule

Zhang Li-Ming1,2, Ouyang Dan-Tong1,2, Zhao Jian1,2 and Bai Hong-Tao1,2,3
1 School of Computer Science and Technology, Jilin University, Changchun 130012, China
2 Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University
3 Center for Computer Fundamental Education, Jilin University, Changchun 130012, China
Email Address: ouyd@jlu.edu.cn; limingzhang@jlu.edu.cn

Received: Received Dec. 19, 2010; Revised Feb. 10, 2011; Accepted Apr. 5, 2011
Published online: 1 October 2012

Abstract: After a deep investigation on the maximum terms space of the clause set, the concept of the partial maximum terms space of the clause set, which the maximum terms of the clause set decomposed, is brought forward. By investigating the extension rule, this paper introduces the concept of the satisfiability and the unsatisfiability of the partial maximum terms space, and gives an algorithm determining the satisfiability of a partial space of the maximum terms - algorithm PSER (Partial Semi-Extension Rule). Then, the TP problem is decomposed into several sub-problems independent of each other, which can be solved by the given parallel computing method PPSER (Parallel Partial Semi-Extension Rule).

Keywords: Theorem Proving, Parallel Algorithm, Extension Rule.

1 Introduction

The classical NP-complete problem of TP has seen much interest in not just the theoretical computer science community, but also in areas where practical solutions to this problem which enable significant practical applications1. However, NP-Completeness does not exclude the possibility of finding algorithms that are efficient enough for solving many interesting satisfiability instances. These instances arise from many diverse areas - many practical problems in AI planning2-5, circuit testing6,7 and verification8-10 for instance.

This research has resulted in the development of several TP algorithms that have seen practical success. These algorithms are based on various principles such as Resolution11,12, Search13, Binary Decision Diagrams14, and Extension rules15.

Extension-rule based TP method has commended considerable respect from many related researchers. For example, Murray16,17 has applied the extension rule into the generation of the target language based on the knowledge compilation, and achieved good results. Besides, many researchers applied the extension rule to the model counting problem18, and many amended it so as to applied it into the TP of modal logic19. Still some researchers improved the extension rule, and put forward series of algorithms such as NER, RIER, etc20,21.

This paper is organized as follows. In section 2, the related extension-rule based TP methods are given. In section 3, the parallel TP method based on the Semi-extension rule is presented. The experimental results of comparing the algorithm proposed in this paper with other algorithms are also presented in section 4. Finally, our work of this paper is summarized in the last section.

2 Extension-Rule based Theorem Proving Method

We begin by specifying the notation that will be used in the rest of this paper. We use Ψ to denote a set of clauses in conjunctive normal form (CNF), C to denote a single clause, and M to denote the set of all the atoms that appear in Ψ. The extension rule is defined as follows.

DEFINITION 15. Given a clause C, C ∈ Ψ, D = {C ∨ A, C ∨ ¬A | “A” is an atom, A ∈ M, “A” and “¬A” does not appear in C}, we call the deduction process proceeding from C to D the deduction rule of on C, and call D the result of applying the extension rule of on C.

THEOREM 15. A clause C is logically equivalent to the result of the extension rule D.

This theorem ensured the equivalence between the original clause set and the expanded clause set,
thus extension rule can be regarded as an inference rule.

**DEFINITION 2.** A non-tautology clause is a maximum term on a set \( M \) if and only if it contains all the atoms in \( M \) in either positive form or negative form.

**THEOREM 2.** Given a set of clauses \( \Psi \), let \( M \) be the set of all the atoms in it (\( |M|=m \)). If all the clauses in \( \Psi \) are maximum terms on \( M \), then the clause set \( \Psi \) is unsatisfiable if and only if it contains \( 2^m \) clauses.

Apparently, the set of all the maximum terms consists of all the atoms in it (\( |M|=m \)). Therefore, it is only need to compute the number of distinct maximum terms can be deduced from the clause set that we can determine its satisfiability. In addition, when counting the number of the maximum terms that can be deduced from the clause set, we can use the inclusion-exclusion principle presented below.

**THEOREM 3.** (Inclusion-exclusion principle)

The element number of the union of sets of \( A_1, A_2, \ldots, A_n \) can be compute using the formula below:

\[
|A_1 \cup A_2 \cup \cdots \cup A_n| = \sum_{i=1}^{n} |A_i| - \sum_{i<j}^{n} |A_i \cap A_j| + \cdots + (-1)^n |A_1 \cap A_2 \cap \cdots \cap A_n|
\]

**THEOREM 4.** The intersection of the sets that consist of the maximum terms expanded by two clauses respectively will be empty if and only if these two clauses contain complementary literals.

Given a set of clauses \( \Psi = \{C_1, C_2, \ldots, C_n\} \), let \( M \) be the set of atoms that appear in it (\( |M|=m \)). Let \( P_i \) be the sets of all the maximum terms we can get from \( C_i \) by using the extension rule, and let \( S \) be the number of distinct maximum terms we can get from \( \Psi \). By using the extension rule, we will have \( S = |P_1 \cup P_2 \cup \cdots \cup P_n| \).

### 3 The Parallel Theorem Proving Algorithm Based on Semi-Extension Rule

The idea of the parallel semi-extension rule based algorithm is as follows. Firstly, the algorithm decomposes the maximum terms space of the clause set into several partial maximum terms spaces, which convert the SAT problem of the clause set into the SAT problem of the partial maximum terms spaces. If there is a certain partial maximum terms space that is satisfiable, then the clause set is satisfiable. If all the partial maximum terms spaces are unsatisfiable, then the clause set is unsatisfiable. In other words, the clause set is satisfiable. In the following, the concept of the partial maximum terms space will be given.

**DEFINITION 3.** For the set \( M = \{L_1, L_2, \ldots, L_m\} \), the \( 2^m \) maximum terms corresponding to \( M \) is \( \{\neg L_1 \lor \neg L_2 \lor \cdots \lor \neg L_m \lor L_1 \lor \neg L_2 \lor \cdots \lor \neg L_m \lor L_1 \lor L_2 \lor \cdots \lor L_m\} \), and we number each maximum term as \( \text{mi}(0), \text{mi}(1), \ldots, \text{mi}(2^m), \text{mi}(2^m)\).

**DEFINITION 4.** Given a clause set \( \Psi = \{C_1, C_2, \ldots, C_n\} \), let \( M \) be the set of its literals, and \( |M|=m \). We call maximum terms space of \( M \) as \( \text{MI}(M) \). Assuming that \( 1 \leq 2^{m} \leq 2^n \), if we would like to decompose the maximum terms space into \( 2^m \) spaces, then each space is of this form \( \text{MIS}(j) = \{\text{mi}(j) \in \{\text{mi}(2^{m0}), \ldots, \text{mi}(2^{m (2^j)}), \ldots, \text{mi}(2^{m (2^{m-1})})\}, 1 \leq j \leq 2^m \} \).

**DEFINITION 5.** For the partial maximum terms space \( \text{MIS}(j) \), \( 1 \leq j \leq 2^m \). If all the maximum terms in it can be expanded by the clauses of the clause set, then \( \text{MIS}(j) \) is said to be unsatisfiable. If there exist a certain maximum term that cannot be expanded by any clause of the clause set, then \( \text{MIS}(j) \) is said to be satisfiable.

**THEOREM 5.** If every partial maximum terms space is unsatisfiable, then the clause set is unsatisfiable. If there is a certain partial maximum terms space that is satisfiable, then the clause set is satisfiable.

In the following, the algorithm PSER which determines the satisfiability of the partial maximum terms space will be given.

**DEFINITION 6.** Let \( M = \{L_1, L_2, \ldots, L_m\} \), \( m \in \mathbb{N} \). Let clause \( C = L_1 \lor \cdots \lor L_i \lor \cdots \lor L_d \), \( 1 \leq i \leq d \leq m \), which \( d \) is referred as the degree of clause \( C \). \( \Psi = \{C \text{V} \text{Li}, C \text{V} \text{Lid}\} \), we call the operation proceeding from \( C \) to the elements of \( \Psi \) the semi-extension rule, and the elements of \( \Psi \) the result of the semi-extension rule.

**PROPOSITION 1.** According to definition 6, when applying the semi-extension rule on \( C \), the remaining \( m-d \) atoms could be positive or negative, therefore \( C \) can semi-expand \( 2^{m-d} \) clauses.

**PROPOSITION 2.** Let \( d_1 \) and \( d_2 \) be the degrees of clause \( C_1 \) and \( C_2 \) respectively, while \( d_1 < d_2 \) and \( C_1 \subseteq C_2 \). According to proposition 1, the clause that \( C_1 \) or \( C_2 \) can semi-expand is obtained by compose the \( m-d_1 \) atoms or \( m-d_2 \) atoms in positive form or in negative form. Therefore, clauses that \( C_2 \) can semi-expand are a subset of the clauses that \( C_1 \) can expand.

According to proposition 2, when determining whether the maximum terms clause can be expanded by the clauses, we should determine whether it can be expanded by the clauses of smaller degree first. In the following, the algorithm determining the satisfiability of the partial maximum terms space will be given.

**Function PSER(CNF; \Psi, INT:starti, INT:endi) :**

1. BEGIN
2. i=starti; \( \Psi=\text{DegreeSort}(\Psi) \);
3. While (i<endi)
4. BEGIN
5. \( T \leftarrow \text{getMaxTerm}(i) \);
6. IF Expand(\( \Psi, T \))=false
7. THEN Return SAT;
8. ELSE If (len<mi)=LastMi(C,M)
9. i++; END
10. END
11. Return UNSAT;
12. END

In the following, the related theorem and algorithm of Expand will be given.

**THEOREM 6.** Given a clause set \( \Psi = \{C_1, C_2, \ldots, C_n\} \), Let \( M \) be the set of its literals, and \( |M|=m \). A maximum term \( T = L_1 \lor L_2 \lor \cdots \lor L_m \) on \( M \) can be expanded by
clause $C= L_1 \lor \ldots \lor L_j \ldots \lor L_d \land 1\leq j \leq d \leq m$, iff \{ $L_1, L_2, \ldots, L_d$ \} $\subseteq \{ L_1, L_2, \ldots, L_m \}$.

In the above, we gave the solving method of partial maximum terms space and the algorithm determining its satisfiability. The maximum terms space of a clause set can be decomposed into several partial maximum terms spaces. In doing so, the SAT problem of a clause set is converted into the SAT problems of several partial maximum terms spaces. If there is a partial maximum terms space that cannot be expanded, then the clause set is unsatisfiable. Or else, if all the partial maximum terms space is unsatisfiable, then the clause set is unsatisfiable.

In the following, the parallel TP algorithm based on semi-extension rule will be given.

Function PPSER(CNF: $\Psi$, INT: threadnum)
1 While $i < \text{threadnum}$ do
2 BEGIN
3 $\text{tid} = \text{createthread}()$;
4 If ($\text{tid} = 0$)
5 BEGIN
6 Result[$i$] = PPSER($\Psi$, start($i$), end($i$));
7 END
8 $i++$
9 END
10 END
11 While (1) do
12 BEGIN
13 BEGIN
14 int Count = 0;
15 BEGIN
16 if (Result[$j$]==SAT)
17 BEGIN
18 return SAT;
19 END
20 END
21 if (Result==unsat) return unsat;
22 END
23

The concrete flow of the algorithm is as follows: The parent process distributes Threadnum sub-threads, and then these sub-threads are arranged to many cores of the processor by the Operating System, respectively. Each sub-thread calls the function PPSER, and records the corresponding returned results using Result[$j$], while the parent process monitors running result of each sub-thread. If there is a sub-thread that its Result[$j$] is SAT, then the algorithm returns SAT. If the Result[$j$] of every sub-thread is UNSAT, then the algorithm returns UNSAT.

4 Experimental Results

On the basis of literature 21, in this section, we compare our algorithm PPSER with algorithm NER, algorithm IER and Directional Resolution algorithm 12 proposed by Dechter and et al, respectively. The experiments are carried out on a Dell Dimension C521, AMD Athlon(tm) 64 X2 Dual Core Processor 3600+, 1.9GHz 1022MB RAM with Windows XP. This experiment uses Uniform Random-3-SAT benchmark 23 lays on phase change zone and standard test cases of frb 24 as test cases. For the entire 1000 issues of uf20-90, table 1 only shows the experimental results of 10 issues randomly selected. The first column of the table shows the sample name, the latter 3 columns shows the runtime of three algorithms corresponding to each issue which the unit of data is Seconds (s). For each test case, we will test 10 times and take the mean value as the experimental results. As we can see from the test, our algorithm PPSER has a significant advantage on efficiency compared with the original algorithm, which is 8-20 times higher than the relatively fast algorithm NER.

Table 1. Experimental Results of Uniform Random-3-SAT Benchmark Instances.

<table>
<thead>
<tr>
<th>PROBLEM</th>
<th>NER(s)</th>
<th>DR(s)</th>
<th>IER(s)</th>
<th>PPSER(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>uf20-02</td>
<td>0.015</td>
<td>1.938</td>
<td>0.062</td>
<td>0.0008</td>
</tr>
<tr>
<td>uf20-05</td>
<td>0.015</td>
<td>1.359</td>
<td>0.109</td>
<td>0.0012</td>
</tr>
<tr>
<td>uf20-07</td>
<td>0.031</td>
<td>0.781</td>
<td>1.218</td>
<td>0.0018</td>
</tr>
<tr>
<td>uf20-09</td>
<td>0.078</td>
<td>4.797</td>
<td>0.609</td>
<td>0.0045</td>
</tr>
<tr>
<td>uf20-018</td>
<td>0.078</td>
<td>1.563</td>
<td>6.968</td>
<td>0.0041</td>
</tr>
<tr>
<td>uf20-023</td>
<td>0.046</td>
<td>1.203</td>
<td>0.484</td>
<td>0.0025</td>
</tr>
<tr>
<td>uf20-036</td>
<td>0.031</td>
<td>1.797</td>
<td>0.313</td>
<td>0.0018</td>
</tr>
<tr>
<td>uf20-040</td>
<td>0.046</td>
<td>1.390</td>
<td>0.453</td>
<td>0.0023</td>
</tr>
<tr>
<td>uf20-042</td>
<td>0.015</td>
<td>2.343</td>
<td>0.250</td>
<td>0.0012</td>
</tr>
<tr>
<td>uf20-069</td>
<td>0.078</td>
<td>4.000</td>
<td>0.484</td>
<td>0.0025</td>
</tr>
</tbody>
</table>

First, we select instances, which the parameters are $\langle N, 20, 10 \rangle$ and $\langle N, 30, 10 \rangle$ respectively, for testing, where the parameter $N$ is restricted as $60 \leq N \leq 160$. For each instance of different difficulty levels, they will randomly generate 10 samples for solving, and let the mean value as the final result. The experimental result is in Figure 1 and Figure 2.

![Fig.1. $\langle N, 20, 10 \rangle$](image)

![Fig.2. $\langle N, 30, 10 \rangle$](image)

By testing the Random SAT problems, we can see from figure 1 and figure 2, that our algorithm PPSER has an obvious advantage on efficiency, which is 6-15 times higher than the relatively fast algorithm NER. Moreover, when the number of clauses increased to 130 above, the
computing time of our algorithm PPSER is increased gently.

5 Conclusions

This method decomposes the maximum terms space of the clause set into several partial maximum terms spaces, and determines the satisfiability of each partial maximum terms space. If all the partial maximum terms space are unsatisfiable, the clause set is unsatisfiable. Or else, if there is a certain partial maximum terms space that is satisfiable, then the clause set is satisfiable. Besides, when determining the satisfiability of the partial maximum terms space, if we found that a maximum term can be expanded by a certain clause, then all the maximum terms semi-expanded based on this clause have no need to determine their expandability, thus effectively reduced the number of the maximum terms to be determined. The experiment results show that our algorithm PPSER has a reasonable execution efficiency, which is superior to algorithm NER, DR, IER and et al.

Acknowledgments

This work was supported in part by NSFC under Grant Nos. 60973089, 60873148, 60773097, 61003101; Jilin Province Science and Technology Development Plan under Grant Nos.20101501, 20100185, 2009010, 20080107, 201101039; Doctoral Fund of Ministry of Education of China(20100061110031);Opening Fund of Top Key Discipline of Computer Software and Theory in Zhejiang Provincial Colleges at Zhejiang Normal University;Zhejiang Provincial Natural Science Foundation under Grant Nos.Y1100191;

References