

# A DPLL Algorithm with Literal Renaming Strategy

Tianyan Deng, Changyun Liu and Daoyun Xu<sup>+</sup>

Department of Computer Science, Guizhou University, Guiyang 550025, China

(Received April 10, 2007, accepted July 22, 2007)

**Abstract.** Restricted on the proof of unsatisfiability of unsatisfiable formulas, a modified *DPLL* — *RSMLS* algorithm is presented. The new algorithm has a symmetry rule (*Literal renaming*) and three simplified rule (*(1,\*)-Resolution, Subformula, and Multiple*). As an example, *RSMLS* algorithm is applied for the proof of unsatisfiability of pigeon-hole formula  $P_{n-1}^n$ . We show with respect to *RSMLS* algorithm that  $P_{n-1}^n$  has a refutation tree with at most  $O(n^3)$  nodes.

**Keywords:** unsatisfiable formula; *DPLL* algorithm; symmetry rule; hard example.

## 1. Introduction

A literal is a propositional variable or a negated propositional variable. For a formula  $F$ ,  $var(F)$  denotes the set of variables occurring in the formula  $F$ ,  $lit(F)$  denotes the set of literals generated by  $var(F)$ . A renaming of formula  $F$  is a mapping  $\varphi$  on  $var(F)$ , satisfy that: for every variable  $x \in var(F)$ ,  $\varphi(x) = x$  or  $\varphi(x) = \neg x$ . A variable renaming of formula  $F$  is a permutation of variables. A literal renaming of formula  $F$  is a combination of a renaming and a variable renaming. Formally, a literal variable renaming of formula  $F$  can be defined by a mapping  $\varphi: lit(F) \rightarrow lit(F)$ , satisfy that  $\varphi(\neg L) = \neg\varphi(L)$ , where  $L$  is a literal, and supposed that  $\neg\neg L = L$ . We can know that: a formula will not change its satisfiability on the action of a literal variable renaming. When we are proving to the satisfiability of a formula, we can proof one formula of two if one formula can change to another formula in the proof process, and eliminating the proof of the other formula. This is the rule of symmetry for propositional proof systems. In *DPLL* algorithm, if a formula labeled to some node and the formula can change to another formula of another node by literal renaming, then we can label one of the two nodes with empty clause. Such, we could reduce the node number of the splitting tree and simplify the proof.

In [1], Krishnamurthy have explained the action of symmetry rule in proof system. However Urquhart have given a disabled example with respect to symmetry rule[2]. Even though when we prove some hard examples of resolution, the symmetry rule still has key action in the proof. If we ignore the times of deciding symmetry we could reduce some hard examples of exponential verify times to polynomial times. Certainly, the idealistic result is that: for a given hard example formula, in the process of proof, verifying symmetry can be finished in polynomial times and proving to formula with symmetry proof system can be finished in polynomial times. Such, the whole process of proof can be finished in polynomial times.

When we use symmetry rule in the proof of propositional calculation, we will meet the following deciding problem: for given *CNF* formulas  $H$  and  $F$ , whether there is a literal renaming  $\varphi$  such that  $\varphi(H)=F$ ? This problem is closely related to the graph isomorphism. In [3,4], we have proved that: even to Horn formulas, this problem is same hard as the graph isomorphism. We know that: until now, the complexity of the problem of graph isomorphism is still a unsolved problem.[5]. It is easy to prove that the deciding problem of graph isomorphism is a NP problem. But we do not know either whether it can be decided in polynomial times or whether it is NP complete problem. So, we consider the literal renaming restricted on subgroups of some minimal unsatisfiable formulas [6], and found some subgroups of minimal unsatisfiable formulas which can be decided in polynomial times if we restricted literal renaming on them. A formula  $F$  is called minimal unsatisfiable if  $F$  is unsatisfiable but the formula which deleted any clause from  $F$  is satisfiable. In [7], C. H. Papadimitriou and D. Wolfe have proved that: for every formula  $F$ , it can be transformed to a another formula in polynomial times, such that

<sup>+</sup> Corresponding author: Phn: 86-851-3627649, dyxu@gzu.edu.cn

- $F$  is satisfiable if and only if  $f(F)$  is satisfiable;
- $F$  is unsatisfiable if and only if  $f(F)$  is minimal unsatisfiable.

From the above we can know that a unsatisfiable formula can be transformed to a minimal unsatisfiable formula in polynomial times. Therefore, it is of great significance to use minimal unsatisfiable formulas to study unsatisfiable formulas. The formulas that we known as resolution hard examples are almost minimal unsatisfiable formulas[1,8,10,11].

In the research of *SAT* problem, we interest to consider the following two kinds of problems: one is how to look quickly for a true assignment to satisfy a satisfiable formula and what characters these formula possessed. Another is how to look quickly for a refutation proof to a unsatisfiable formula and what characters these formula possessed.

In this paper, based on the idea due to H. Kleine Büning and Zhao X.S.[12], we restricted on the proof of unsatisfiability of unsatisfiable formulas. We consider an improved strategy with a literal renaming of the well known *DPLL* algorithm[4]. Meanwhile, we introduce the following three simplified rules: (1) (1,\*)-resolution (it is permit to use (1,\*)-resolution in the algorithm); (2) subformula rule(it is permit to delete extra clauses in the algorithm); (3) multiple rule (we can replace one of the same formulas with empty clause). By introduce the above three rules, we given out an improved algorithm from *DPLL* algorithm—called *RSMLS* algorithm. As a applied example, we will apply *RSMLS* algorithm to the proof of unsatisfiability the famous hard formula—pigeon hole formula  $P_{n-1}^n$ . We have proved that: based on *RSMLS* algorithm, the formula  $P_{n-1}^n$  there is a refutation tree with at most  $O(n^3)$  nodes.

## 2. Preliminaries

In this section we will introduce some necessity definitions, notations and some results.

A clause  $C$  is a disjunction of literals( $C=L_1 \vee L_2 \vee \dots \vee L_n$ ), or a set of literals( $C=\{L_1, L_2, \dots, L_n\}$ ). A formula  $F$  in conjunctive normal form (*CNF*) is a conjunction of clauses,  $F=(C_1 \wedge C_2 \wedge \dots \wedge C_m)$ , or a set  $\{C_1, C_2, \dots, C_m\}$  of clauses, or a list  $[C_1, C_2, \dots, C_m]$  of clauses. A subformula of a given formula  $F$  is a formula which consists of part clauses of the formula  $F$ .

If a clause contains a pair of propositional and negated of a same variable, it is called a tautology clause. In a formula, we can delete the tautology clauses and it will not affect the satisfiability of a formula. A clause is called unit clause if it contained only one literal. A variable  $x$  is called positive (negative) occurring in a formula  $F$  if literal  $x$  ( $\neg x$ ) occurring in formula  $F$ .

We denote  $\#var(F)$  as the number of variables occurring in  $F$  and  $\#cl(F)$  as the number of clauses in  $F$ .  $\#cl(F)-\#var(F)$  is called the deficiency of a formula  $F$ , denoted by  $d(F)$ .

Davis-Putnam-Loveland-Logemann (*DPLL*) algorithm is a very important algorithm to solve the *SAT* problem, the detailed description about *DPLL* algorithm can be found in [13,14,17]. In general *DPLL* algorithm, the following 4 rules are applied to simplify or resolute the problem:

**1. tautology clause :** Delete tautology clauses from formula  $F$ .

**2. unit clause :** If formula  $F$  contained a unit clause  $L$ , then delete  $L$  and delete all  $\neg L$  in all other clauses which contain  $\neg L$ .

**3. pure literal :** If a literal  $L$  occurring in some clauses of the formula  $F$  and literal  $\neg L$  not occurring in all rest clauses of the formula  $F$ , then delete the all clauses which contained  $L$ .

**4. splitting :** If the formula  $F$  is the form as:  $F=[(x \vee f_1), \dots, (x \vee f_m), (\neg x \vee g_1), \dots, (\neg x \vee g_n), h_1, \dots, h_l]$ , where clauses  $f_1, \dots, f_m, g_1, \dots, g_n, h_1, \dots, h_l$  do not contain  $x, \neg x$ , then the problem to verify the satisfiability of formula  $F$  can be split into two sub problems: to verify the satisfiability of  $F(x=0)=[f_1, \dots, f_m, h_1, \dots, h_l]$  and  $F(x=1)=[g_1, \dots, g_n, h_1, \dots, h_l]$ . If anyone of them is satisfiable then  $F$  is satisfiable, otherwise  $F$  is unsatisfiable.

It is not difficult to prove that: In the process of use *DPLL* to verify the satisfiability of a formula, If only use the rules of true rule, unit rule and pure literal rule, then this kind of formula can complete in polynomial times. However, in general, it is unavoidable to use the splitting rule in the process of use *DPLL* algorithm. Therefore, the splitting rule is the core of *DPLL* algorithm. The use of the splitting rule will generate to two parts and the other rules will produced only a simplified problem. So we can use a binary

tree to denote intuitively the process of the use of *DPLL* algorithm. Such binary tree is called the splitting tree of *DPLL* algorithm or simplified as the splitting tree. The root node of the splitting tree is labelled by the initial formula  $F$ . Every step the use of the rules will generate one or two new successor nodes, the new nodes are labelled by the new formula relatively. The leaf of the splitting tree is labelled by empty formula or empty clause. If all leafs are labelled by empty clauses, the initial formula  $F$  is unsatisfiable. Otherwise,  $F$  is satisfiable.

In the process of apply *DPLL* algorithm, the nodes number of the splitting tree is a measure form of process complexity to verify a formula. The various improved strategies to *DPLL* algorithm are how to introduce new rules and combination of rules for reducing the nodes number. In [17], the author has had a very good sum-up to some famous improved strategies.

**Definition 1.** A renaming of a formula  $F$  is a mapping  $\varphi$  from  $lit(F)$  to  $lit(F)$ , such that for every variable  $x$ ,  $\varphi(x) \in \{x, \neg x\}$ , where  $lit(F)$  is the set of literals in  $F$ .

A variable renaming of a formula  $F$  is a permutation over the set of variables in  $F$ .

A literal renaming of a formula  $F$  is a mapping  $\varphi$  from  $lit(F)$  to  $lit(F)$ , such that for every literal  $L$ ,  $\varphi(\neg L) = \neg\varphi(L)$  (supposed  $\neg\neg L = L$ ).

In fact, a literal renaming is a combination of a renaming and a variable renaming.

**Definition 2.** Suppose  $F = [(L \vee f), (\neg L \vee g_1), \dots, (\neg L \vee g_m), F_{rest}]$  is a formula, where  $L$  is a literal,  $f, g_1, \dots, g_m$  are clauses,  $F_{rest}$  is a formula which not occurring the literal  $L$  and  $\neg L$ , we call the formula  $F' = [(f \vee g_1), \dots, (f \vee g_m), F_{rest}]$  as  $(1, *)$ -resolution about literal  $L$  of  $F$ .

It is obvious that unit resolution is a particular case of  $(1, *)$ -resolution.

**Definition 3.** Suppose  $F = [C_1, C_2, \dots, C_n]$  is a formula,  $F$  is called as a minimal unsatisfiable formula (simplified as *MU*) if  $F$  is unsatisfiable and  $F \setminus C_i$  be satisfiable for any clause  $C_i (1 \leq i \leq n)$  in  $F$ .

It has proved that the difference  $\#cl(F) - \#var(F)$  of a minimal unsatisfiable formula  $F$  is a positive integer [15,16]. In general, we can group the minimal unsatisfiable formulas depends on the difference  $\#cl(F) - \#var(F)$ . For positive  $k \geq 1$ , denoted  $MU(k) = \{F \in MU \mid \#cl(F) - \#var(F) = k\}$ , the formula in  $MU(k)$  is called  $MU(k)$  formula.

**Lemma 1.** Suppose  $F = [(L \vee f), (\neg L \vee g_1), \dots, (\neg L \vee g_m), F_{rest}]$  is a formula, where  $L$  is a literal,  $f, g_1, \dots, g_m$  are clauses,  $F_{rest}$  is a formula which not occurring the literal  $L$  and  $\neg L$ .  $F' = [(f \vee g_1), \dots, (f \vee g_m), F_{rest}]$  is  $(1, *)$ -resolution about literal  $L$  of  $F$ , then  $F$  is  $MU(k)$  formula if and only if  $F'$  is  $MU(k)$  formula.

Proof : without loss of generality, suppose  $F = [(x \vee f), (\neg x \vee g_1), \dots, (\neg x \vee g_p), F_{rest}]$ , where  $F_{rest}$  not occurring  $x$  and  $\neg x$ , so  $F' = [(f \vee g_1), \dots, (f \vee g_p), F_{rest}]$ .

( $\Rightarrow$ ) suppose  $F \in MU(k)$  and  $\#var(F) = n$ , then  $\#cl(F) = n+k$ ,  $\#var(F') = n-1$  and  $\#cl(F') = n+k-1$ . so  $d(F') = k$ . we will prove  $F' \in MU(k)$ .

(1)  $F'$  is unsatisfiable. If not so, there is a true assignment  $v$ , such that  $v(F') = 1$ . It satisfied  $v(f \vee g_1) = \dots = v(f \vee g_p) = v(F_{rest}) = 1$ . The following is an extended assignment  $v'$  from  $v$ , such that  $v'(F) = 1$ :

$$v'(y) = \begin{cases} v(y) & y \in var(F) \setminus \{x\} \\ 0 & y = x \text{ and } v(f) = 1 \\ 1 & y = x \text{ and } v(f) = 0 \end{cases}$$

It is clear that  $v'(F) = 1$ . It is contradiction with the unsatisfiability of  $F$ .

(2)  $F'$  is a minimal unsatisfiable formula. Otherwise, there is a clause  $h \in F'$  such that  $F' \setminus \{h\}$  is unsatisfiable. There are the following cases.

Case 1.1 for some  $i (1 \leq i \leq p), h = (f \vee g_i)$

Without loss of generality, suppose  $h = (f \vee g_1)$ . Since  $F$  is a minimal unsatisfiable formula, we have that  $F_1 = [(x \vee f), (\neg x \vee g_2), \dots, (\neg x \vee g_p), F_{rest}]$  is satisfiable. So there is a true assignment  $v$  such that  $v(F_1) = 1$ . This implies  $v(x \vee f) = v(\neg x \vee g_2) = \dots = v(\neg x \vee g_p) = v(F_{rest}) = 1$ . If  $v(x) = 1$ , then  $v(g_2) = \dots = v(g_p) = v(F_{rest}) = 1$ . If  $v(x) = 0$ , then  $v(f) = v(F_{rest}) = 1$ . So, we have  $v(F' \setminus \{h\}) = 1$ . It is contrary.

Case 1.2  $h \in F_{rest}$

As  $F$  is minimal unsatisfiable,  $F_1 = [(x \vee f), (-x \vee g_1), \dots, (-x \vee g_p), F'_{rest}]$  is satisfiable, where  $F'_{rest} = F_{rest} \setminus \{h\}$ . So there is a true assignment  $\nu$  such that  $\nu(F_1)=1$ . This implies  $\nu(x \vee f) = \nu(-x \vee g_1) = \dots = \nu(-x \vee g_p) = \nu(F'_{rest}) = 1$ . If  $\nu(x)=1$ , then  $\nu(g_1) = \dots = \nu(g_p) = \nu(F'_{rest}) = 1$ . If  $\nu(x)=0$ , then  $\nu(f) = \nu(F'_{rest}) = 1$ . So, we have  $\nu(F \setminus \{h\}) = 1$ . It is contrary.

( $\Leftarrow$ ) Suppose  $F' = [(f \vee g_1), \dots, (f \vee g_p), F_{rest}] \in MU(k)$ . We will prove that  $F \in MU(k)$ .

First,  $F$  is unsatisfiable. Otherwise, there is a true assignment  $\nu$  such that  $\nu(F)=1$ . If  $\nu(x)=1$ , then  $\nu(g_1) = \dots = \nu(g_p) = \nu(F_{rest}) = 1$ . If  $\nu(x)=0$ , then  $\nu(f) = \nu(F_{rest}) = 1$ . So, we have  $\nu(F') = 1$ . It is contrary.

Second,  $F$  is minimal unsatisfiable, otherwise, there is a clause  $h \in F$  such that  $F \setminus \{h\}$  is unsatisfiable. There are the following cases.

Case 2.1  $h = (x \vee f)$

Note that:  $F_{rest}$  is satisfiable. So there is a true assignment  $\nu$  such that  $\nu(F_{rest})=1$ . We can expand  $\nu$  to be another true assignment  $\nu'$  such that  $\nu'(F \setminus \{h\}) = 1$ :

$$\nu'(y) = \begin{cases} \nu(y) & y \in \text{var}(F) \setminus \{x\} \\ 0 & y = x \end{cases}$$

Case 2.2 for some  $i(1 \leq i \leq p), h = (-x \vee g_i)$

Without loss of generality, suppose  $h = (-x \vee g_1)$ . Since  $F'$  is a minimal unsatisfiable formula, we have that:  $F_1 = [(f \vee g_2), \dots, (f \vee g_p), F_{rest}]$  is a satisfiable formula. So, there is a true assignment  $\nu$  such that  $\nu(F_1)=1$ . it is the same as above, we can expand the assignment  $\nu$  to  $\nu'$  such that  $\nu'(F \setminus \{h\}) = 1$ :

$$\nu'(y) = \begin{cases} \nu(y) & y \in \text{var}(F) \setminus \{x\} \\ 0 & y = x \text{ and } \nu(f) = 1 \\ 1 & y = x \text{ and } \nu(f) = 0 \end{cases}$$

Case 2.3  $h \in F_{rest}$

As  $F'$  is minimal unsatisfiable, so  $F_1 = [(f \vee g_1), \dots, (f \vee g_p), F'_{rest}]$  is satisfiable, where  $F'_{rest} = F_{rest} \setminus \{h\}$ . So there is a true assignment  $\nu$  such that  $\nu(F_1)=1$ . Using the same expanded method in case 2.2, we will get a true assignment  $\nu'$  from  $\nu$ , such that  $\nu'(F \setminus \{h\}) = 1$ . It is contrary. This is end of the proof.

Lemma 1 shows that the formula in  $MU(k)$  is closed in (1,\*)-resolution, that is, a minimal unsatisfiable formula is still a minimal unsatisfiable formula after once (1,\*)-resolution, and the differences of the number of clauses and the number variables are equal.

For  $n > 1$ , we denoted the pigeon-hole principle with  $n$  pigeons and  $n-1$  holes to a formula  $PHP_{n-1}^n$  as follows.

$$PHP_{n-1}^n = \bigwedge_{1 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1}) \rightarrow \bigvee_{1 \leq h \leq n-1} \bigvee_{1 \leq i < j \leq n} (x_{i,h} \wedge x_{j,h}).$$

From the above formula  $PHP_{n-1}^n$ , we define the pigeon-hole formula  $P_{n-1}^n$  with  $n$  pigeon and  $n-1$  holes as follows (simplified the pigeon-hole formula).

$$P_{n-1}^n = \bigwedge_{1 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1}) \wedge \bigwedge_{1 \leq h \leq n-1} \bigwedge_{1 \leq i < j \leq n} (\neg x_{i,h} \vee \neg x_{j,h}).$$

In formula  $P_{n-1}^n$ , variable  $x_{i,j}$  denotes the  $i$ th pigeon be put into the  $j$ th hole. Clearly,  $\text{var}(P_{n-1}^n) = n(n-1)$  and  $cl(P_{n-1}^n) = n + \frac{1}{2}n(n-1)^2$ .

Proposition 1:  $P_{n-1}^n$  is a minimal unsatisfiable formula.

Proof: (1)  $P_{n-1}^n$  is unsatisfiable.

Suppose the formula  $\bigwedge_{1 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1}) \wedge \bigwedge_{1 \leq h \leq n-1} \bigwedge_{1 \leq i < j \leq n} (\neg x_{i,h} \vee \neg x_{j,h})$  is satisfiable. There is a true assignment  $\varphi$  over the variable set  $\{x_{p,h} \mid 1 \leq p \leq n, 1 \leq h \leq n-1\}$ , such that:

$$(1.1) \text{ For every } p(1 \leq p \leq n), \varphi(x_{p,1} \vee \dots \vee x_{p,n-1}) = 1.$$

(1.2) For every pair  $i, j(1 \leq i < j \leq n)$ , and every  $h(1 \leq h \leq n-1)$ ,  $\varphi(\neg x_{i,h} \vee \neg x_{j,h}) = 1$ .

From (1.1), for every  $p(1 \leq p \leq n)$ , there is some  $h(1 \leq h \leq n-1)$ , such that  $\varphi(x_{p,h}) = 1$ . By pigeon principle: there exists some pair  $i, j(1 \leq i < j \leq n)$ , and some  $h(1 \leq h \leq n-1)$ , such that  $\varphi(x_{i,h}) = \varphi(x_{j,h}) = 1$ . This is contradiction with (1.2).

Intuitively, the satisfiability of the formula  $\bigwedge_{1 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1})$  expressed that the  $n$  pigeons have to be put into the holes. The satisfiability of the formula  $\bigwedge_{1 \leq h \leq n-1} \bigwedge_{1 \leq i < j \leq n} (\neg x_{i,h} \vee \neg x_{j,h})$  expressed that: in the  $n$  pigeons, every pair pigeons can not be put into the same hole. Now there are only  $n-1$  holes so it is not possible to satisfy the both formulas.

(2)  $P_{n-1}^n$  is minimal unsatisfiable.

That is, we need to prove that: The formula that we delete any clause  $C$  from  $P_{n-1}^n$  is satisfiable. We discuss them as the following two cases.

Case 1 For some  $p(1 \leq p \leq n)$ ,  $C = (x_{p,1} \vee \dots \vee x_{p,n-1})$ .

Without loss of generality, we suppose  $C = (x_{1,1} \vee \dots \vee x_{1,n-1})$ . Deleting  $C$  from the formula  $P_{n-1}^n$ , we get the formula:

$$F_1 = \bigwedge_{2 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1}) \wedge \bigwedge_{1 \leq h \leq n-1} \bigwedge_{1 \leq i < j \leq n} (\neg x_{i,h} \vee \neg x_{j,h}).$$

We define a true assignment  $\Psi_1$  over the variables set  $\{x_{p,h} \mid 1 \leq p \leq n, 1 \leq h \leq n-1\}$  as follows:

$$\Psi_1(x) = \begin{cases} 0 & x \in \{x_{1,1}, \dots, x_{1,n-1}\} \\ 1 & x \in \{x_{2,1}, \dots, x_{n,n-1}\} \\ 0 & x \in \{x_{p,h} \mid 1 \leq p \leq n, 1 \leq h \leq n-1\} - \{x_{1,1}, \dots, x_{1,n-1}, x_{2,1}, \dots, x_{n,n-1}\} \end{cases}$$

so, we have that  $\Psi_1(F_1) = 1$ .

Intuitively, deleting  $C = (x_{1,1} \vee \dots \vee x_{1,n-1})$  from the formula  $P_{n-1}^n$ , it expressed that the No. 1 pigeon may be not put into the hole. Then we could put the No. 2 pigeon into the No. 1 hole, put the No. 3 pigeon into the No. 2 hole, ..., the No.  $n$  pigeon into the No.  $n-1$  hole. This is what the true assignment  $\Psi_1$  denoted.

Case 2 : For some pair  $i, j(1 \leq i < j \leq n)$ , and for some  $h(1 \leq h \leq n-1)$ ,  $C = (\neg x_{i,h} \vee \neg x_{j,h})$ .

Without loss of generality, we suppose  $C = (\neg x_{1,1} \vee \neg x_{2,1})$ . Deleting clause  $C$  from the formula  $P_{n-1}^n$ , we get the following formula:

$$F_2 = \bigwedge_{1 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1}) \wedge \bigwedge_{2 \leq h \leq n-1} \bigwedge_{1 \leq i < j \leq n} (\neg x_{i,h} \vee \neg x_{j,h}) \wedge \bigwedge_{2 \leq i < j \leq n} (\neg x_{i,1} \vee \neg x_{j,1}) \wedge \bigwedge_{3 \leq j \leq n} (\neg x_{1,1} \vee \neg x_{j,1}).$$

We define a true assignment  $\Psi_2$  over variables set  $\{x_{p,h} \mid 1 \leq p \leq n, 1 \leq h \leq n-1\}$  as the following:

$$\Psi_2(x) = \begin{cases} 1 & x \in \{x_{1,1}, x_{2,1}\} \\ 1 & x \in \{x_{3,2}, \dots, x_{n,n-1}\} \\ 0 & x \in \{x_{p,h} \mid 1 \leq p \leq n, 1 \leq h \leq n-1\} - \{x_{1,1}, x_{2,1}, x_{3,2}, \dots, x_{n,n-1}\} \end{cases}$$

Then, we have that  $\Psi_2(F_2) = 1$ .

Intuitively, if we delete the clause  $C = (\neg x_{1,1} \vee \neg x_{2,1})$  from the formula  $P_{n-1}^n$ , it expressed that No.1 and No.2 pigeons can both be put into No.1 hole. This is what the true assignment  $\Psi_2$  denoted.

From the above, the formula  $P_{n-1}^n$  is a minimal unsatisfiable formula. It is the end of the proof.

Furthermore, we have that  $P_{n-1}^n \in MU(\frac{1}{2}n(n^2 - 4n + 5))$ .

### 3. Improved Algorithm

The improved algorithm is based on the idea due to H. Kleine Büning and Zhao X.S [12].

We consider an extension of the well known *DPLL* algorithm. The new algorithm is called *RSMLS* algorithm (Resolution-Subformula-Literal renaming-Multiple-Splitting). *RSMLS* algorithm is restricted on the refutation proof of unsatisfiable formulas. In this algorithm, resolution rule is specified as (1,\*)-resolution, subformula rule is specified as a unsatisfiable subformula from a unsatisfiable formula, symmetry rule is literal renaming, multiple rule is only choose one from the same formulas, splitting rule is the same as in *DPLL* algorithm.

With *DPLL* algorithm similarly, we still use the form of tree to descript the using process of the rules. The tree we obtained is still called splitting tree. In the splitting tree, we still use a formula to label a node. On the leaf nodes, it is permitted to use blank clause ( $\square$ ) to label them.

In the *RSMLS* algorithm, we use the following rules.

#### 1. Initiating

The root is labeled with the initial formula  $F_0$ .

#### 2. (1,\*)-Resolution

Let  $v$  be a leaf with the label  $F = [(L \vee f), (\neg L \vee g_1), \dots, (\neg L \vee g_r), F_{rest}]$ , where neither  $L$  nor  $\neg L$  occurs in  $F_{rest}$ . Then  $v$  has one successor node  $v_c$  with label  $[(f \vee g_1), \dots, (f \vee g_r), F_{rest}]$ . (Note: unit resolution is particular case of (1,\*)-resolution)

#### 3. Subformula

Let  $v$  be a leaf with the label formula  $F$ , and  $F_c$  is a non-empty subformula of  $F$ , then  $v$  has one successor node  $v_c$  with label  $F_c$ . (Note: When this rule be used,  $F_c$  should be a unsatisfiable formula.)

#### 4. Literal renaming

Let  $v$  be a leaf with the label formula  $F$ , and  $\varphi$  is a literal renaming of  $F$ , such that  $\varphi(F) = F_c$ , then  $v$  generates one successor node  $v_c$  with label  $F_c$ .

#### 5. Multiple

Let  $v$  be a leaf with the label formula  $F$ . If there exist some other node  $v'$  with label  $F$ , then we add a successor node  $v_c$  labeled by the empty clause  $\square$ .

#### 6. Splitting

Let  $v$  be a leaf with the label formula  $F$ . and the literal  $L$  occurs both positively and negatively in  $F$ , then  $v$  has two successor nodes  $v_l$  and  $v_r$ , the node  $v_l$  is labeled with  $F(L=1)$  and the other node  $v_r$  is labeled with  $F(L=0)$ , where  $F(L=1)$  (or  $F(L=0)$ ) is simplified by replace  $L$  with 1 (or 0) in  $F$ .

#### 7. End

If all leafs are labeled with the empty clause  $\square$ , then we halt with return the value contradictory.

It is easy to prove that a formula  $F$  is unsatisfiable if and only if there is a *RSMLS* refutation with return the value contradictory.

In *RSMLS* algorithm, the subformula rule implied the true rule in *DPLL* algorithm.

## 4. The application of *RSMLS* algorithm

In this section, as an applied example, we will apply *RSMLS* algorithm to prove the unsatisfiability of the famous hard example—pigeon hole formula  $P_{n-1}^n$ , and we also prove that the  $P_{n-1}^n$  has a refutation tree with at most  $O(n^3)$  nodes.

For convenience to read, we repeat the pigeon holes formulas as follows.

$$P_{n-1}^n = \bigwedge_{1 \leq p \leq n} (x_{p,1} \vee \dots \vee x_{p,n-1}) \wedge \bigwedge_{1 \leq h \leq n-1} \bigwedge_{1 \leq i < j \leq n} (\neg x_{i,h} \vee \neg x_{j,h}) .$$

From the proposition 1, we know that  $P_{n-1}^n$  is a minimal unsatisfiable formula and  $P_{n-1}^n \in MU(\frac{1}{2}n(n^2 - 4n + 5))$ .

**Theorem 1.** For formula  $P_{n-1}^n$ , there exists a refutation tree based on *RSMLS* algorithm, and its splitting tree has at most  $O(n^3)$  nodes.

**Proof:** In the proof we define the following formulas and notations.

$$H_{L,R} = \{(\neg x_{i,k} \vee \neg x_{j,k}) : i, j \in L, i < j \text{ and } k \in R\}, \text{ (Using clause set to express formula, the rest is the same.)}$$

$$G_{L,R} = \{(\vee_{k \in R} x_{i,k} \vee \vee_{k \in R} x_{j,k}) : i, j \in L \text{ and } i < j\},$$

$$Q_{L,R} = H_{L,R} + G_{L,R}. \text{ ("+" express the union of sets, the rest is the same.)}$$

where  $L, R \subseteq N$  (That is, the subsets of natural number set) with  $|L|=l, |R|=r$ , and  $1 \leq r < l$ .

For  $i, j, l \in N$  with ( $j \leq l$ ), we denote that  $X_{i,j-l} = (x_{i,j} \vee x_{i,j+1} \vee \dots \vee x_{i,l})$ . Similarly, for a set  $S \subseteq N$  and  $i \in N$ , we denote that  $X_{i,S} = \vee_{j \in S} x_{i,j}$ .

Thus, we have that  $G_{L,R} = \{(X_{i,R} \vee X_{j,R}) : i, j \in L \text{ and } i < j\}$ , and

$$P_{n-1}^n = \{X_{i,1-(n-1)} : 1 \leq i \leq n\} + \{(\neg x_{i,k} \vee \neg x_{j,k}) : 1 \leq i < j \leq n \text{ and } 1 \leq k \leq n-1\}$$

$$= \{(x_{i,1} \vee X_{i,2-(n-1)}) : 1 \leq i \leq n\} + \{(\neg x_{i,k} \vee \neg x_{j,k}) : 1 \leq i < j \leq n \text{ and } 1 \leq k \leq n-1\}.$$

Now, we can begin apply *RSMLS* to construct the refutation tree of the formula  $P_{n-1}^n$ .

(A<sub>1</sub>) We label root node of the splitting tree with  $P_{n-1}^n$ . Now the tree only contains one node.

Chose  $P_{n-1}^n$ , we apply (1,\*)-resolution in turn to every variable of  $x_{1,1}, x_{2,1}, \dots, x_{n,1}$  (it is used n times). The generated formula is that:  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} = G_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} + H_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ , where

$$H_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} = \{(\neg x_{i,k} \vee \neg x_{j,k}) : 1 \leq i < j \leq n \text{ and } 2 \leq k \leq n-1\}, G_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} = \{(X_{i,2-(n-1)} \vee X_{j,2-(n-1)}) : 1 \leq i < j \leq n\}.$$

The formula  $G_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  can be explained as No.1 hole is closed.

Since  $P_{n-1}^n$  is a minimal unsatisfiable formula, by lemma 1 we have got that  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  is a minimal unsatisfiable formula.

Until now, the splitting tree has only one leaf node and label it with formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ .

(A<sub>2</sub>) For formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ , we consider splitting on the variables  $x_{1,2}, x_{1,3}, \dots, x_{1,n-1}$  in turn firstly. The generated splitting tree as the following graph: (where the leaf nodes labeled with nodes label, and the branch nodes labeled with their label formula.)

In the above splitting process, we have the following cases.

Case 1:  $x_{1,2} = 1$ , Its branch node labels with  $v_{1,2}^{(1)}$ .

Case 2: For some  $d(2 \leq d \leq n-2)$ ,  $x_{1,2} = x_{1,3} = \dots = x_{1,d} = 0$  and  $x_{1,d+1} = 1$ . Its branch node labels with  $v_{1,d+1}^{(1)}$ .

Case 3:  $x_{1,2} = x_{1,3} = \dots = x_{1,n-1} = 0$ . Its branch node labels with  $v_{1,n-1}^{(0)}$ .

Now we deal with the above 3 kinds of cases respectively. Suppose  $n > 2$ , what the formula we concern to is  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ .

**Case 1.**  $x_{1,2} = 1$

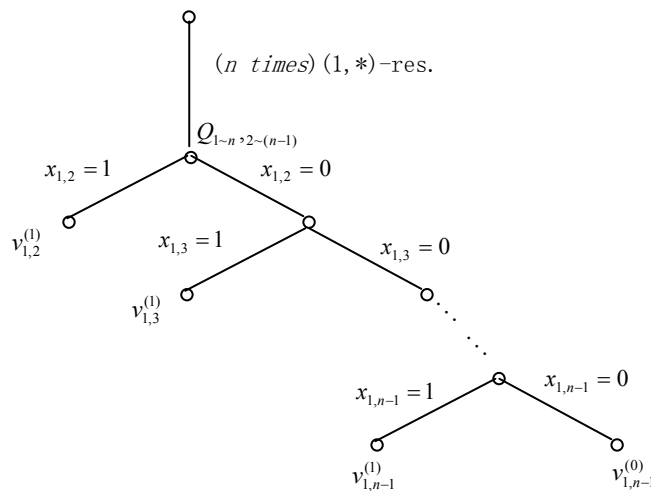


Fig. 1 Part splitting tree

(1.1) Firstly, we compute  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=1)$ , and the simplified formula labels with  $v_{1,2}^{(1)}$ .

Since  $x_{1,2}=1$ , we delete the clauses which occur the literal  $x_{1,2}$  and delete the literal  $\neg x_{1,2}$  in the rest clauses which occur the literal  $\neg x_{1,2}$  in  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ . The generated formula is as follows.

$$Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=1) = G'+H',$$

where

$$\begin{aligned} G' &= \{X_{i,2-(n-1)} \vee X_{j,2-(n-1)} : 2 \leq i < j \leq n\}, \\ H' &= \{\neg x_{j,2} : 2 \leq j \leq n\} + \{(\neg x_{i,2} \vee \neg x_{j,2}) : 2 \leq i < j \leq n\} + \\ &\quad \{(\neg x_{i,k} \vee \neg x_{j,k}) : 2 \leq i < j \leq n \text{ and } 3 \leq k \leq n-1\} + \{(\neg x_{1,k} \vee \neg x_{j,k}) : 2 \leq j \leq n \text{ and } 3 \leq k \leq n-1\}. \end{aligned}$$

Since  $G'+H'$  contains the unit clause set  $\{\neg x_{j,2} : 2 \leq j \leq n\}$ , so we could simplify further: deleting the clause set  $\{(\neg x_{i,2} \vee \neg x_{j,2}) : 2 \leq i < j \leq n\}$  in the formula  $G'+H'$ .

Then, we get the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=1) = G''+H''$ , where

$$\begin{aligned} G'' &= \{X_{i,2-(n-1)} \vee X_{j,2-(n-1)} : 2 \leq i < j \leq n\} \\ H'' &= \{\neg x_{j,2} : 2 \leq j \leq n\} + \{(\neg x_{i,k} \vee \neg x_{j,k}) : 2 \leq i < j \leq n \text{ and } 3 \leq k \leq n-1\} + \\ &\quad \{(\neg x_{1,k} \vee \neg x_{j,k}) : 2 \leq j \leq n \text{ and } 3 \leq k \leq n-1\}. \end{aligned}$$

(1.2) In formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=1) = G''+H''$ , it contains the unit clause set  $\{\neg x_{j,2} : 2 \leq j \leq n\}$ . We apply respectively the (1,\*)-resolution to the unit clause  $\neg x_{j,2} (2 \leq j \leq n)$  (in fact, it is unit resolution), total use  $n-1$  times. The generated formula is the following.

$$Q_{\{2,3,\dots,n\}\{3,4,\dots,n-1\}} + \{(\neg x_{1,k} \vee \neg x_{j,k}) : 2 \leq j \leq n \text{ and } 3 \leq k \leq n-1\}.$$

(1.3) using once the subformula rule to delete the redundant clause set  $\{(\neg x_{1,k} \vee \neg x_{j,k}) : 2 \leq j \leq n \text{ and } 3 \leq k \leq n-1\}$ , we get a minimal unsatisfiable formula  $Q_{\{2,3,\dots,n\}\{3,4,\dots,n-1\}}$ .

(1.4) for formula  $Q_{\{2,3,\dots,n\}\{3,4,\dots,n-1\}}$ , using once the literal renaming rule (in fact, it only need to make a permutation over the subscripts), we get the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ .

Until now, we have got the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$  from the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=1)$ . We have applied the rules  $n+1$  times. So, in the branch of  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=1)$ , it have generated  $(n+2)$  nodes.

**Case2.** for some  $d (2 \leq d \leq n-2)$   $x_{1,2} = x_{1,3} = \dots = x_{1,d} = 0$  and  $x_{1,d+1} = 1$ .

The method is similar to the case 1.

(2.1) First, we compute that  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=x_{1,3}=\dots=x_{1,d}=0, x_{1,d+1}=1)$ , and use the simplified formula to label the node  $v_{1,d+1}^{(1)}$ .

Since  $x_{1,2}=x_{1,3}=\dots=x_{1,d}=0$ , and  $x_{1,d+1}=1$ , we delete the clauses which occur any literal of  $\neg x_{1,2}, \neg x_{1,3}, \dots, \neg x_{1,d}, x_{1,d+1}$  in the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  and delete all the literal  $x_{1,2}, x_{1,3}, \dots, x_{1,d}, \neg x_{1,d+1}$  from the rest clauses which occur any literal of  $x_{1,2}, x_{1,3}, \dots, x_{1,d}, \neg x_{1,d+1}$ . We have got the formula as follows:

$$Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=x_{1,3}=\dots=x_{1,d}=0, x_{1,d+1}=1) = G'+H',$$

where

$$\begin{aligned} G' &= \{X_{i,2-(n-1)} \vee X_{j,2-(n-1)} : 2 \leq i < j \leq n\}, \\ H' &= \{\neg x_{j,d+1} : 2 \leq j \leq n\} + \{(\neg x_{i,d+1} \vee \neg x_{j,d+1}) : 2 \leq i < j \leq n\} + \{(\neg x_{1,k} \vee \neg x_{j,k}) : 2 \leq j \leq n \text{ and } d+2 \leq k \leq n-1\} + \\ &\quad \{(\neg x_{i,k} \vee \neg x_{j,k}) : 2 \leq i < j \leq n, 2 \leq k \leq n-1 \text{ and } k \neq d+1\}. \end{aligned}$$

As there are unit clause set  $\{\neg x_{j,d+1} : 2 \leq j \leq n\}$  in the formula  $G'+H'$ , so it can be further simplified, that is, deleting the clause set of  $\{(\neg x_{i,d+1} \vee \neg x_{j,d+1}) : 2 \leq i < j \leq n\}$  from the formula  $G'+H'$ .

So, we have got the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}(x_{1,2}=x_{1,3}=\dots=x_{1,d}=0, x_{1,d+1}=1) = G''+H''$ ,

Where

$$G'' = \{X_{i,2-(n-1)} \vee X_{j,2-(n-1)} : 2 \leq i < j \leq n\} ,$$

$$H'' = \{-x_{j,d+1} : 2 \leq j \leq n\} + \{(-x_{1,k} \vee -x_{j,k}) : 2 \leq j \leq n \text{ and } d+2 \leq k \leq n-1\} +$$

$$\{(-x_{i,k} \vee -x_{j,k}) : 2 \leq i < j \leq n, 2 \leq k \leq n-1 \text{ and } k \neq d+1\} .$$

(2.2) In the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} (x_{1,2} = x_{1,3} = \dots = x_{1,d} = 0, x_{1,d+1} = 1) = G'' + H''$ , it contained the unit clause set  $\{-x_{j,d+1} : 2 \leq j \leq n\}$ . We apply the (1,\*)-resolution to these unit clauses  $-x_{j,d+1} (2 \leq j \leq n)$  respectively (in fact, this is unit resolution), and it is used n-1 times. The generated formula is the following.

$$Q_{\{2,3,\dots,n\}\{2,3,\dots,d,d+2,\dots,n-1\}} + \{(-x_{1,k} \vee -x_{j,k}) : 2 \leq j \leq n \text{ and } d+2 \leq k \leq n-1\} .$$

(2.3) Using the subformula rule once to delete the redundant clause set  $\{(-x_{1,k} \vee -x_{j,k}) : 2 \leq j \leq n \text{ and } d+2 \leq k \leq n-1\}$ , we will get the minimal unsatisfiable formula  $Q_{\{2,3,\dots,n\}\{2,3,\dots,d,d+2,\dots,n-1\}}$ .

(2.4) Using the literal renaming rule once to the formula  $Q_{\{2,3,\dots,n\}\{2,3,\dots,d,d+2,\dots,n-1\}}$  (That is, making a permutation over the subscripts of variables), we will get the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ .

From the above process of the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  to  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ , we could consider them making two steps as follows.

**First step:** computing the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} (x_{1,2} = x_{1,3} = \dots = x_{1,d} = 0)$  (using the splitting rule  $d$  times), the formula we get denoted by  $Q'$ .

**Second step:** computing the formula  $Q'(x_{1,d+1} = 1)$ , and making the above (2.1)—(2.4) steps to deal with them. It similar to the first step from  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} (x_{1,2} = 1)$  to  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ . We have used (n+1) rules. So, in the branch of  $Q'(x_{1,d+1} = 1)$ , it has generated (n+2) nodes.

(2.5) in case 1, we have got the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ . So, here we can use the multiple rule once to generate a successor node and label it with blank clause  $\square$ .

Then, in the branch of  $Q'(x_{1,d+1} = 1)$ , there are (n+3) nodes.

**Case 3.**  $x_{1,2} = x_{1,3} = \dots = x_{1,n-1} = 0$ .

(3.1) First, we compute the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} (x_{1,2} = x_{1,3} = \dots = x_{1,n-1} = 0)$  and label the node  $v_{1,n-1}^{(0)}$  with the simplified formula.

As  $x_{1,2} = x_{1,3} = \dots = x_{1,n-1} = 0$ , we delete the clauses which occur the literal  $-x_{1,2}, -x_{1,3}, \dots, -x_{1,n-1}$  from the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  and delete the literals  $x_{1,2}, x_{1,3}, \dots, x_{1,n-1}$  from the rest clauses which occur any literal of  $x_{1,2}, x_{1,3}, \dots, x_{1,n-1}$ . We will get the formula as follows.

$$Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}} (x_{1,2} = x_{1,3} = \dots = x_{1,n-1} = 0) = G' + H' ,$$

where

$$G' = \{X_{j,2-(n-1)} : 2 \leq j \leq n\} + \{X_{i,2-(n-1)} \vee X_{j,2-(n-1)} : 2 \leq i < j \leq n\} ,$$

$$H' = \{(-x_{i,k} \vee -x_{j,k}) : 2 \leq i < j \leq n \text{ and } 2 \leq k \leq n-1\} .$$

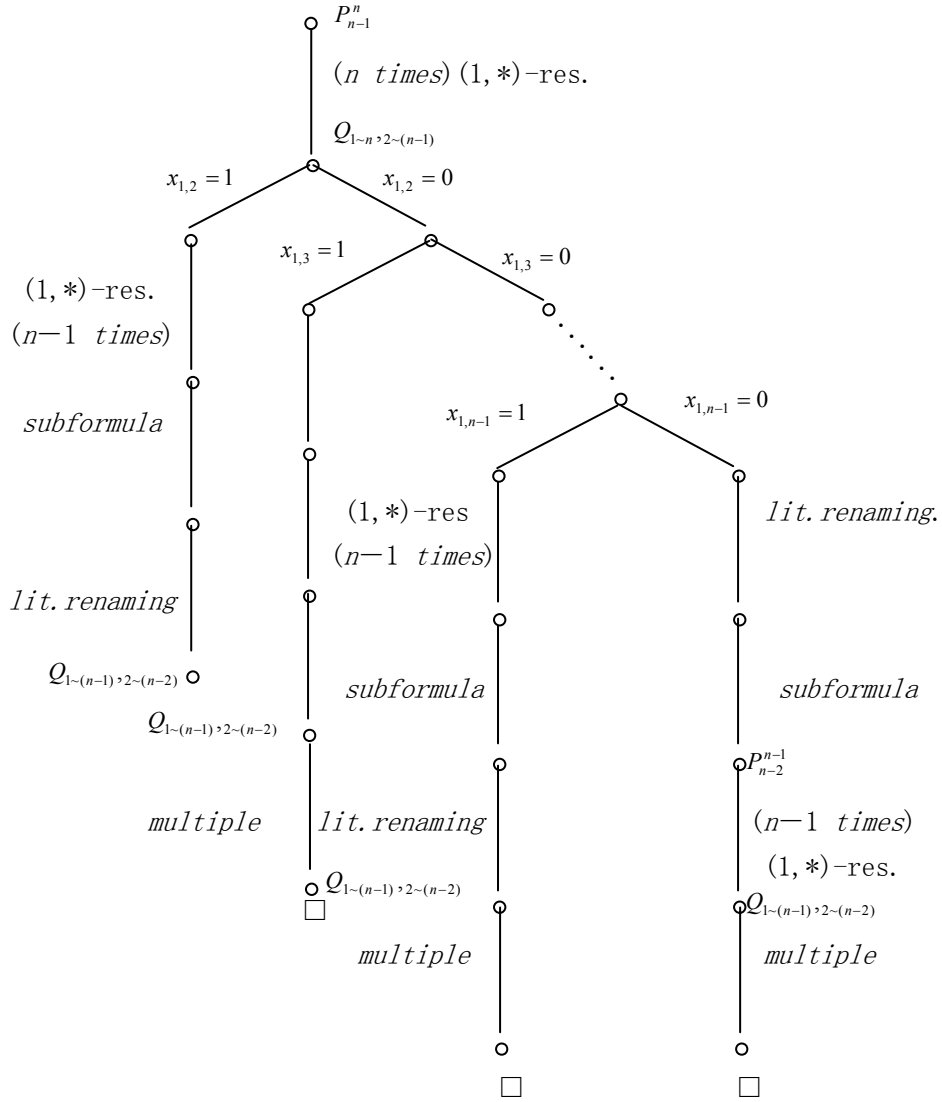


Fig. 2 Splitting tree

(3.2) Using the literal renaming rule once to formula  $G'+H'$  (that is, making a permutation over the subscripts), we get the formula as follows.

$$\begin{aligned}
 Q' &= \{X_{j,1-(n-2)} : 1 \leq j \leq n-1\} + \{(-x_{i,k} \vee -x_{j,k}) : 1 \leq i < j \leq n-1 \text{ and } 1 \leq k \leq n-2\} + \\
 &\quad \{X_{i,1-(n-2)} \vee X_{j,1-(n-2)} : 1 \leq i < j \leq n-1\} \\
 &= P_{n-2}^{n-1} + \{X_{i,1-(n-2)} \vee X_{j,1-(n-2)} : 1 \leq i < j \leq n-1\} .
 \end{aligned}$$

(3.3) Using the subformula rule to the above formula  $Q'$ , we will get the minimal unsatisfiable formula  $P_{n-2}^{n-1}$ .

(3.4) Using the similar method from  $P_{n-1}^n$  to  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ , we can use the (1,\*)-resolution  $n-1$  times to the formula  $P_{n-2}^{n-1}$ , and get the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ .

(3.5) In case 1, we get the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ . So, we can use multiple rule once to the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$  and generate a successor node which we labeled with the blank clause  $\square$ .

So, in the branch of  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  ( $x_{1,2} = x_{1,3} = \dots = x_{1,n-1} = 0$ ), there are  $(n+3)$  nodes.

From the above, we get the splitting tree as Fig. 2.

Now, we compute the node-number of the tree in Fig. 2.

(1) From  $P_{n-1}^n$  to  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ , it generated n new nodes.

(2) In the subtree that its root is  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$ , the node number is  $T(n)=(n+3)+(n-3)(n+4)+(n+3)=(n-1)(n+3)+(n-3)$ . Then the node number of the tree in graph 2 is  $f(n)=n+T(n)$ . Note that  $T(n)$  is a quadric polynomial.

In graph 2, all leaf nodes are labeled by empty clause ( $\square$ ) except the most left node is labeled by the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ .

(3) Now we evaluate the node number of the refutation tree (splitting tree) of  $P_{n-1}^n$  which constructed similarly with the method mentioned above.

In graph 2, the most left node is labeled by the formula  $Q_{\{1,2,\dots,n-1\}\{2,3,\dots,n-2\}}$ . Using the method of (A<sub>2</sub>) which we deal with the formula  $Q_{\{1,2,\dots,n\}\{2,3,\dots,n-1\}}$  similarly until get the formula  $Q_{\{1,2,3\}\{2\}}$ . Then, the node number of the splitting tree is that

$$n+T(n)+T(n-1)+\dots+T(4)-(n-4)$$

(where, (n-4) is the root number of subtree which be computed twice.)

Note that:

$$\begin{aligned} Q_{\{1,2,3\}\{2\}} &= \{x_{1,2} \vee x_{2,2}, x_{1,2} \vee x_{3,2}, x_{2,2} \vee x_{3,2}, \neg x_{1,2} \vee \neg x_{2,2}, \neg x_{1,2} \vee \neg x_{3,2}, \neg x_{2,2} \vee \neg x_{3,2}\}, \\ Q_{\{1,2,3\}\{2\}}(x_{1,2}=1) &= \{x_{2,2} \vee x_{3,2}, \neg x_{2,2}, \neg x_{3,2}, \neg x_{2,2} \vee \neg x_{3,2}\}, \\ Q_{\{1,2,3\}\{2\}}(x_{1,2}=0) &= \{x_{2,2}, x_{3,2}, x_{2,2} \vee x_{3,2}, \neg x_{2,2} \vee \neg x_{3,2}\}. \end{aligned}$$

To the formula  $Q_{\{1,2,3\}\{2\}}(x_{1,2}=1)$ , we apply the subformula rule once, get the minimal unsatisfiable formula  $\{x_{2,2} \vee x_{3,2}, \neg x_{2,2}, \neg x_{3,2}\}$ , then apply (1,\*)-resolution twice to it (in fact, it is unit resolution), get the blank clause  $\square$ .

To the formula  $Q_{\{1,2,3\}\{2\}}(x_{1,2}=0)$ , we apply the subformula rule once, get the minimal unsatisfiable formula  $\{x_{2,2}, x_{3,2}, \neg x_{2,2} \vee \neg x_{3,2}\}$ , then apply (1,\*)-resolution twice to it (in fact, it is unit resolution), get the blank clause  $\square$ .

From the above we can know that: only 8 new nodes be generated by dealing with  $Q_{\{1,2,3\}\{2\}}$  to blank clause  $\square$ .

In summary, to the pigeon-hole formula  $P_{n-1}^n$ , there exists a refutation tree which based on the RSMLS algorithm. And the node number of the refutation(splitting) tree is  $T(n)+T(n-1)+\dots+T(4)+12$ . As  $T(n)$  is a quadric polynomial, so  $T(n)+T(n-1)+\dots+T(4)+12 = O(n^3)$ . That is, the node number of the refutation tree constructed above is at most  $O(n^3)$ . It is end of the proof.

## 5. The Conclusion and Further Works

On the refutation proof of unsatisfiable formulas in this paper, we have introduced literal renaming rule, (1,\*)-resolution rule, subformula rule and multiple rule (here the literal renaming is symmetry rule and other three rules are mainly used to simplify the formula and the proof) in DPLL algorithm. So a modified DPLL-RSMLS algorithm is presented. RSMLS algorithm is effective for some formulas which have symmetry constructions and can essentially improve the hard of the proof. As an example, RSMLS algorithm is applied for the proof of unsatisfiability of the famous pigeon-hole formula  $P_{n-1}^n$ . We prove that: based on RSMLS algorithm,  $P_{n-1}^n$  has a refutation tree with at most  $O(n^3)$  nodes. We should note that: to prove a given formula, the literal renaming rule is the key step to improve the complexity of the proof. So, if the deciding problem of literal renaming between two formulas can be finished in polynomial times, the application of RSMLS algorithm will be obviously more effective. The further work is to consider that some kinds of formulas which literal renaming deciding problem can be finished in polynomial times.

Thanks. We owe special thanks to Prof. Hans Kleine Büning (Computer and Science department of Paderborn University, German) and Prof. Zhao X.S. (the Institute of Logic and Cognition, Sun Yat-Sen University), for their support, help, and guidance.

## 6. References

- [1] K. Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*. 1985, **22**: 253-275.
- [2] A. Urquhart. The Symmetry Rule in Propositional Logic. *Discrete Applied Mathematics*. 1999, **96-97**: 177-193.
- [3] H. Kleine Büning, D.Y. Xu.. The complexity of homomorphisms and renamings for minimal unsatisfiable formulas. *Conference SAT2002*, Cincinnati.
- [4] D.Y. Xu. *On the Complexity of Renamings and Homomorphisms for Minimal Unsatisfiable Formulas*. Dissertation, Nanjing University in China, 2002 .
- [5] J. Köbler, U. Schöning, J. Toran. *The graph isomorphism problem: its structural complexity*. Birkhäuser Verlag, 1993.
- [6] D.Y. Xu. Polynomial decidability of renaming for formulas in MU(1). *Journal of Guizhou University*. 2003, **20**(1): 9-19.
- [7] C. H. Papadimitriou, D. Wolfe. The complexity of facets resolved. *Journal of Computer and System Sciences*. 1988, **37**: 2-13.
- [8] A. Atserias, N. Galesi, R. Gavaldà. Monotone proofs of the pigeon hole principle. In: U. Montanari et al. (eds.): *ICALP 2000*, LNCS 1853, Berlin Heidelberg: Springer-Verlag, 151-162.
- [9] V. Chvatal, T. Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*. 1988, **35**: 759-768.
- [10] A. Haken. The intractability of resolution. *Theoretical Computer Science*. 1985, **39**: 297-308.
- [11] A. Urquhart. Hard examples for resolution. *Journal of ACM*, 1987, **34**: 209-219.
- [12] H. Kleine Büning, X. S. Zhao. Minimal unsatisfiability: results and open questions. *Conference SAT2002*, Cincinnati.
- [13] M. Davis, G. Logemann, D. Loveland. A machine program for theorem proving. *Communications of the ACM*. 1962, **7**(5): 394-397.
- [14] M. Davis, H. Putnam. A computing procedure for quantification theory. *Journal of ACM*. 1960, **7**(3): 201-215.
- [15] G. Davydov, I. Davydova, H. Kleine Büning. An efficient algorithm for the minimal unsatisfiability problem for a subclass of CNF. *Annals of Mathematics and Artificial Intelligence*. 1998, **23**: 229-245 .
- [16] H. Kleine Büning. On subclasses of minimal unsatisfiable formulas. *Discrete Applied Mathematics*. 2000, **107**: 83-98.
- [17] Zhang Jian. *The Decision of Satisfiability for Logic Formulas-Methods, Tools and Applications*, Science Press, 2000, 16-28. (Chinese)