A Partition Rule for SAT Solvers: The Multiple Partition Rule (MPR)

Juan Segura-Salazar¹, Juan Frausto-Solís²

¹Genomic Science Center, UNAM, P.O. Box 565-A, Av. Universidad s/n, C.P. 62210, Cuernavaca Morelos, México.
²ITESM, Cuernavaca Campus, A.P. 99-C, Paseo de la Reforma 182-A, C.P. 62589, Cuernavaca Morelos, México.

Abstract

We propose a new partition rule for DPLL-based SAT Solvers. Most of the complete SAT solvers usually are based on Davis, Logemann and Loveland (DPLL) rules. One most DPLL rule actually used in the modern algorithms is the Classical Partition Rule (CPR), that divides the problem into sub-problems (resolvents) and thereby it finds a solution through a decision tree. In this paper a new partition rule named Multiple Partition Rule (MPR) is presented. MPR generates a new decision tree according to clauses instead CPR which generates a decision tree according to variables. MPR can be used for developing new SAT’s algorithms and to improve existence ones that use CPR. Experimental results comparing MPR versus CPR show that using MPR makes more efficient solutions than CPR.

Keywords: Propositional satisfiability problem, SAT, Np-Complete, Algorithms.

1. Introduction

All Since Cook proved that the propositional satisfiability problem (SAT) [1] was the first problem shown to be NP-Complete [2], many other problems have been considered in this category [3]. However, SAT remains at the core of this classification due: i) to its simple representation (this does not mean that SAT can be solved easily), and ii) it has been shown that many other NP-Complete problems can be transformed to a SAT instance in polynomial time [4].

Owing to its importance, many SAT algorithms have been developed based either on local search (incomplete) or backtrack search (complete) [5]-[6]. It is striking that most popular and useful complete algorithms for solving SAT are based on Davis, Logemann and Loveland algorithm (DPLL) rules [7]. DPLL [8] is a modified version of the algorithm proposed by Davis & Putnam in 1960 [7]. The DPLL corresponds to a backtrack search; the Boolean Constraint Propagation (BCP) use the Classical Partition Rule (CPR) [9] to make a decision tree to order the solution and apply backtracking. Modern SAT solvers like zchaff [10] and Grasp [11] have efficient BCP engines for detection of unit clauses, propagation unit literals and conflict detection. This solvers improve the basic backtrack search of DPLL implementing a non-chronology backtrack and conflict analysis, but using the basic BCP to partition the problem. In this sense, given the importance of CPR in the execution of SAT solvers, it is very important to create alternative rules in order to get more efficient SAT algorithms based in CPR rule.

This paper introduces a new partition rule named Multiple Partition Rule (MPR) which focuses on solving clauses in contrast to CPR that solves variables, and generates a new decision tree in the SAT propagation process. To demonstrate the efficiency of MPR, it was implemented on an algorithm named MPR_Solver which involves a different partition mechanism, and CPR it was implemented on an algorithm named CPR_Solver. Results show that MPR’s efficiency outperforms CPR’s efficiency in fifty percent. The paper organization is as follows: in the next section basic definitions are provided, in section 3 the Classical Partition Rule (CPR) [7] is analyzed and discussed. In section 4, the Multiple Partition Rule (MPR) is presented and its application in a SAT algorithm named MPR_Solver to compare with CPR_Solver is discussed in Section 5.

2. Basic Definitions

To represent an instance of SAT, the next elements are required:

- A set of variables which can be false (0) or true (1), \( \beta = x_1, \ldots, x_n \)
- A set of literals \( L \). A literal is a variable that can be denied or not denied, this is:

\[
\exists x \in \beta: (L = x) \lor (L = \neg x)
\]  

(1)
• A set of clauses, \( C = c_1, \ldots, c_m \) where each one is a disjunction of literals, this is:

\[
c_h = \bigvee_{k+1 \leq j \leq m} x_j^i, -1 \leq j \leq 1, 1 \leq h \leq m
\] (2)

Where: \( j = \begin{cases} j^0 = x_j \\ j^1 = \neg x_j \end{cases} \)

• A boolean formula in CNF, this is:

\[
\varphi = \bigwedge_{1 \leq h \leq m} c_h
\] (3)

The CNF formula is also represented by a set of clauses, rather than as a conjunction. For example, an alternative way to represent \( \varphi \) is as the set \( \{C_1, \ldots, C_m\} \).

In this sense, a formal definition of SAT is given next:

**Definition 1.** Find some true assignment for the variables, such that the formula \( \varphi \) is true; otherwise demonstrate that an assignment of values does not exist to evaluate \( \varphi \) as true.

Many SAT instances are limited to the length \( k \); such instances are denoted \( k\text{-SAT} \). The most common instances of special interest are 2-SAT and 3-SAT. It has been proved that for \( k = 2 \) such instances are solved in polynomial time \([12]-[13]\) while for \( k = 3 \) the instances are NP-complete \([2]\). In this paper, we use 3-SAT instances in order to prove the MPR approach.

The propagation for variable \( x \) is denoted by \( \varphi[x] \), that means:

1. If \( x = 1 \), remove \( \neg x \) from \( \varphi \)'s clauses and remove clauses containing \( x \) from \( \varphi \).
2. If \( x = 0 \), remove \( x \) from \( \varphi \)'s clauses and remove clauses containing \( \neg x \) from \( \varphi \).

If \( \varphi \) contains no clauses, then \( \varphi = 1 \) (true). The SAT problem represented by \( \varphi \) its satisfiable if there exist an assignment \( \lambda \), such that \( \varphi = 1 \) under \( \lambda \), otherwise it is unsatisfiable.

An empty clause denoted by \( \Box \) is a clause that does not have literals and always it’s unsatisfiable.

### 3. Classical Partition Rule (CPR)

The DPPL \([7]\) is backtrack search algorithm. Two rules are the base for this algorithm:

1. Classic Partition Rule (CRP), called also splitting Literal Rule:

If the set \( \varphi \) can be expressed in the way:

\[
(x_{i_1} \lor x_{i_2} \lor \cdots \lor x_{i_k}) \land \cdots \land (x_{j_1} \lor x_{j_2} \lor \cdots \lor x_{j_l})
\]

where \( x_{ij} \) is the variable \( j \) of the clause \( i \), then the following set is obtained:

\[
\begin{align*}
\varphi_1 &= x_{i_1} \land (x_{i_2} \lor x_{i_3} \lor \cdots \lor x_{i_k}) \land \cdots \land (x_{j_1} \lor x_{j_2} \lor \cdots \lor x_{j_l}) \\
\varphi_2 &= \neg x_{i_1} \land (x_{i_2} \lor x_{i_3} \lor \cdots \lor x_{i_k}) \land \cdots \land (x_{j_1} \lor x_{j_2} \lor \cdots \lor x_{j_l}) \\
\vdots \\
\varphi_k &= \neg x_{j_k} \land (x_{i_1} \lor x_{i_2} \lor \cdots \lor x_{i_k}) \land \cdots \land (x_{j_1} \lor x_{j_2} \lor \cdots \lor x_{j_l})
\end{align*}
\] (4)

\( \varphi \) is unsatisfiable if and only if \( \varphi_1 \lor \varphi_2 \lor \cdots \lor \varphi_k \) is unsatisfiable; that is, if all are unsatisfiable.

2. Unit Clause Rule (UCR):

If exist a clause \( c_b = x_a \) in \( \varphi \), then \( \varphi[x_a] \), if exist a clause \( c_b = \neg x_a \) in \( \varphi \), then \( \varphi[\neg x_a] \).

CPR to divide the problem into two sets according to a selected variable; one set is evaluated with a true value and the other with a false value. The resulting sets are solved searching for one that is satisfiable or both are unsatisfiables. CPR also orders the solution search by creating a binary decision tree, which eliminates repetition of partial assignments proven previously. A possible decision tree formed by CPR is shown in the Fig. 1.

![Fig. 1: Possible decision tree formed by the CPR.](image-url)
is $2^n$. The first level involves the derivation of two arches, which then form two arches for the next level; the process continues in this way until the last level. Hereof, the total number of arches is:

$$\sum_{i=1}^{n} 2^i = 2^{n+1} - 2 \quad (6)$$

Modern complete SAT solvers implement the CRP on the following way: A partial assignment $\omega$, initialized empty, is maintained. A $\omega$ is extended by a BCP [14]; then, if the input $\phi$ is neither 1 nor 0 under $\omega$, the CRP is used, the satisfiability of $\phi$ is recursively checked under $\{\omega, x\}$ and then under $\{\omega, \neg x\}$, as required. The $x$ is a selected variable that is picked using some decision heuristic.

4. The Multiple Partition Rule (MPR)

In a SAT instance, clauses are formed by several variables joined with the disjunction operator (OR). It means that at least one variable in a clause needs to have a true value (1) to make its respective clause true. It also implicates not assigning the values that will be assigned to its variables avoiding the process according to the clause. For example, the following is an examination of the clause $c_i = (x_1 \lor x_2 \lor x_3)$ of an instance $\phi$ with a literal set $\{x_1, x_2, x_3, x_4, x_5\}$. The clause $c_i$ indicates that any one of $x_1, x_2$ or $x_3$ should take a true value to make this clause true; it also indicates that the assignment $\{\neg x_1, \neg x_2, \neg x_3\}$ with any of the other variables is invalid. The set of invalid assignments is shown in the Fig. 2.

$$\{\neg x_1, \neg x_2, \neg x_3\},$$
$$\{\neg x_1, \neg x_2, \neg x_3, x_4, x_5\},$$
$$\{\neg x_1, \neg x_2, \neg x_3, x_4, \neg x_5\},$$
$$\{\neg x_1, \neg x_2, \neg x_3, x_5, \neg x_4\},$$
$$\{\neg x_1, \neg x_2, \neg x_3, x_4, x_5\},$$
$$\{\neg x_1, \neg x_2, \neg x_3, x_4, \neg x_5\},$$
$$\{\neg x_1, \neg x_2, \neg x_3, x_4, x_5\}$$

Fig. 2: Set of invalid assignments, clause $(x_1 \lor x_2 \lor x_3)$

In the case of the decision tree of CRP shown in Fig. 1, it accepts this invalid assignment despite the CRP is focused on the selected variable. In this sense, the CRP procedure (when the variable $x_i$ has been selected to be propagated, and the variables $x_5, x_3, x_4, x_6$ remain in the clause) can be expressed by:

CRP. If $((\phi[\omega, x_1])\{x_2, x_3, x_4, x_5\} = 0)$ Then

$$((\phi[\omega, \neg x_1])\{x_2, x_3, x_4, x_5\}$$

From (7) it is clear that the CRP procedure divides the instance in two parts looking for the solution in any one of them: one assigning a true value to the selected variable ($x_1$) and other with a false value ($\neg x_1$). It is also clear that invalid assignments will be taken into account in the propagation procedure.

We propose a new partition rule (MPR) which focuses on the clause. MPR considers a clause by transforming the partition process according to the values that will be assigned to its variables avoiding invalid assignments. The MPR function is formed in the following way:

MPR. If $((\phi[\omega, x_1])\{x_2, x_3, x_4, x_5\} = 0)$ Then

$$(\phi[\omega, \neg x_1])\{x_2, x_3, x_4, x_5\}$$

The valid values for the clause ($x_1 \lor x_2 \lor x_3$) are $\{x_1, x_1\}, \{\neg x_1, x_2, x_2\}$ and $\{\neg x_1, x_2, x_2\}$. Using this function repeatedly, the multiple partition rule is intuitively designed as follows:

Multiple Partition Rule (MPR): If the set $\varphi$ can be expressed in the way:

$$(x_{i_1} \lor x_{i_2} \lor \ldots \lor x_{i_n}) \land \ldots \land (x_{n_1} \lor x_{n_2} \lor \ldots \lor x_{n_m}) \quad (9)$$

where $x_{ij}$ is the variable $j$ of the clause $i$, then the following set is obtained:

$$\varphi_1 = x_{i_1} \land (x_{i_2} \lor x_{i_2} \lor x_{i_2}) \land \ldots \land (x_{n_1} \lor x_{n_2} \lor \ldots \lor x_{n_m})$$
$$\varphi_2 = \neg x_{i_1} \land (x_{i_2} \lor x_{i_2} \lor x_{i_2}) \land \ldots \land (x_{n_1} \lor x_{n_2} \lor \ldots \lor x_{n_m})$$
$$\varphi_3 = \neg x_{i_1} \land \neg x_{i_1} \land \ldots \land (x_{i_2} \lor x_{i_2} \lor \ldots \lor x_{i_2}) \land (x_{n_1} \lor x_{n_2} \lor \ldots \lor x_{n_m})$$

$$(10)$$

$\varphi$ is unsatisfiable if and only if $(\varphi_1 \lor \varphi_2 \lor \ldots \lor \varphi_3)$ is unsatisfiable; that is, if all are unsatisfiable.

With this new rule, the variable to be selected considers the result of a previous assignment in the clause. When MPR gives unsatisfiable in the evaluation process for a specific value of a selected variable $x_0$, MPR now considers this variable with the contrary value. After $x_1$ has been assigned with true and false, MPR proceeds to select another variable to spread. This process is shown in Fig. 3.
The application of this rule is an improvement since the number of selections decreases. In the decision tree formed by MPR (Fig. 3), the right arcs fix the value, and not propagation. This means that the decision tree decreases (and so the number of selections) according to:

$$\sum_{i=1}^{n} 2^i / 2 = 2^n - 1$$

(11)

MPR reduces the number of operations for the assignment and propagation in comparison with CPR by 50%.

5. CRP vs. MPR

MPR can be used in the design of new algorithms or improve algorithms CRP based; but in the case of algorithms CRP based, their efficiency it depends on the programming and the philosophy of each one.

Some modern algorithms use clauses for the detection and maintenance conflicts. For this reason, to prove to MRP against CRP basic algorithms were implemented: CRP_Solver which uses CRP and UCR (Fig. 4) and MRP_Solver which uses MRP and UCR (Fig. 5).

To show the MPR performance let us consider an example with the following instance $\phi$ formed by three variables $(x_1, x_2, x_3)$ and eight clauses:

$$\phi = \{(x_1 \lor x_2 \lor x_3), (\sim x_1 \lor x_2 \lor x_3),(x_1 \lor \sim x_2 \lor \sim x_3), (\sim x_1 \lor \sim x_2 \lor x_3), (x_1 \lor \sim x_2 \lor \sim x_3), (\sim x_1 \lor x_2 \lor \sim x_3), (\sim x_1 \lor x_2 \lor x_3)\}$$

(12)

The decision tree formed by CRP is shown in the Fig. 6 while the decision tree formed by MPR is shown in the Fig. 7. It can be observed that with MRP the number of operations is smaller than with CRP.
6. CRP vs. MPR

In order to test MPR_Solver and to compare it with CPR_Solver some instances of SATLIB [15] were considered. The SAT instances were separated into two groups: satisfiable (uf) and unsatisfiable (uf) and all of these were comprised of 50 variables and 218 clauses.

In Table 1 and Table 2, the comparative results of MPR_Solver and CPR_Solver are shown. It can be seen that MPR is always more efficient than CPR.

The obtained results are as follows:

- The results of the uf instances are shown in Fig. 8. It can be seen that MPR reduces the solution time by 30% with respect to CPR (Table 1).
- The results of the uuf instances evaluated with MPR and CPR can be seen in Fig. 9. In these instances, the solution time obtained with MPR was decreased about 66% with respect to CPR (Table 1).

In general, results show that the MPR reduce the solution time of satisfiable and unsatisfiable instances by 57% with respect to CPR (Table 2).

<table>
<thead>
<tr>
<th>Time</th>
<th>MPR uf</th>
<th>CRP uf</th>
<th>MPR uuf</th>
<th>CRP uuf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>1,206.89</td>
<td>4,508.50</td>
<td>1,613.52</td>
<td>1,675.70</td>
</tr>
<tr>
<td>Medium</td>
<td>989.22</td>
<td>2,973.39</td>
<td>706.39</td>
<td>1,014.87</td>
</tr>
<tr>
<td>Maximum</td>
<td>5,535.13</td>
<td>13,294.37</td>
<td>9,931.29</td>
<td>11,836.25</td>
</tr>
<tr>
<td>Minimum</td>
<td>231.00</td>
<td>364.86</td>
<td>4.72</td>
<td>35.78</td>
</tr>
</tbody>
</table>

Table 1: Results of instances uf and uuf using MRP and CRP.

7. Conclusions

In this paper, a new rule named the Multiple Partition Rule (MPR) is presented. After conducting a decision tree analysis, it was demonstrated that MPR is at least fifty percent more efficient than CPR because the maximum number of visited arcs with MPR is 2ⁿ in lieu of 2ⁿ⁺¹ with CPR. This theoretical result was tested through experiments using CRP_Solver and MRP_Solver. The fact is that CPR is used in algorithms based on the DPLL rules like zchaff [10] and GRASP[11], therefore an improvement to CPR could redound to an improvement of the algorithms.
based on her. But even, MPR can be used in the creation of a new type of algorithms.

References


