

A Multi-Processor Approach to SAT-Problems

Christian Posthoff¹, Bernd Steinbach²

¹ The University of the West Indies, St. Augustine Campus,
Trinidad & Tobago

² Freiberg University of Mining and Technology, Germany

Outline

1. Introduction into SAT and TVL
2. Constructive Approach
3. Implementation of the Intersection
4. Parallelization on the Processor Level
5. Reduction of Solution Sets
6. Clause Ordering – “Small” Intermediate Results
7. Experimental Results
8. Conclusions

Introduction

Basic Definitions

$$B^n = \{(b_1, \dots, b_n) \mid b_i \in \{0,1\}, i = 1, \dots, n\}$$

$$V = (x_1, \dots, x_n) \quad \text{n binary variables}$$

$$D = x_1 \vee \dots \vee x_k \quad \text{disjunction of variables or complemented variables (literals)}$$

$$CF = D_1 \wedge \dots \wedge D_l \quad \text{conjunctive form}$$

Introduction

SAT-Problem

The most general SAT-problem:
Find all solutions of this equation!

$$CF = 1$$

Subproblems: Find some solutions!

Confirm that there are solutions!

Confirm that there are no solutions!

SAT is an NP-complete problem!

Introduction

Ternary vectors

$(1 \ - \ 0 \ -)$ represents a set of binary vectors:
 $\{(1 \ 0 \ 0 \ 0), (1 \ 0 \ 0 \ 1), (1 \ 1 \ 0 \ 0), (1 \ 1 \ 0 \ 1)\}$

$\begin{bmatrix} 1 & - & 0 & - \\ 0 & 1 & - & 0 \end{bmatrix}$ A list of ternary vectors can be understood as the union of such sets.

Constructive Approach

Construction of partial solutions

Restrict the considerations to 3-SAT: no loss of generality!

Orthogonal coding of the solution set of one conjunction:

$$C_1 = x_1 \vee \bar{x}_2 \vee x_3$$
$$\begin{pmatrix} 1 & - & - \\ 0 & 0 & - \\ 0 & 1 & 1 \end{pmatrix}$$

Constructive Approach

The intersection of two vectors

$x_i \cap y_i$	0	1	–
0	0	\emptyset	0
1	\emptyset	1	1
–	0	1	–

This operation will be applied to each component.

Constructive Approach

Implementation of the intersection

Coding of the ternary values by two bits:

0	coded by	10
1	coded by	11
–	coded by	00

Two bit vectors represent one ternary vector.

Constructive Approach

Orthogonality of two vectors

There does not exist an intersection vector, if:

$$\mathit{bit} 1(x) \wedge \mathit{bit} 1(y) \wedge (\mathit{bit} 2(x) \oplus \mathit{bit} 2(y)) \neq 0$$

Only three parallel vector operations are necessary – 16, 32, 64 or 128 variables can be considered simultaneously.

Constructive Approach

Parallel calculation of the intersection

When two vectors are not orthogonal then the intersection can be implemented by two parallel bit vector operations:

$$\mathit{bit} 1(x \cap y) = \mathit{bit} 1(x) \vee \mathit{bit} 1(y)$$

$$\mathit{bit} 2(x \cap y) = \mathit{bit} 2(x) \vee \mathit{bit} 2(y)$$

Parallelization on the Processor Level

Decomposition into subtasks

The solution set of the equation

$$C_1 \wedge \dots \wedge C_k = 1$$

can be replaced by the intersection of the solution set of partial equations:

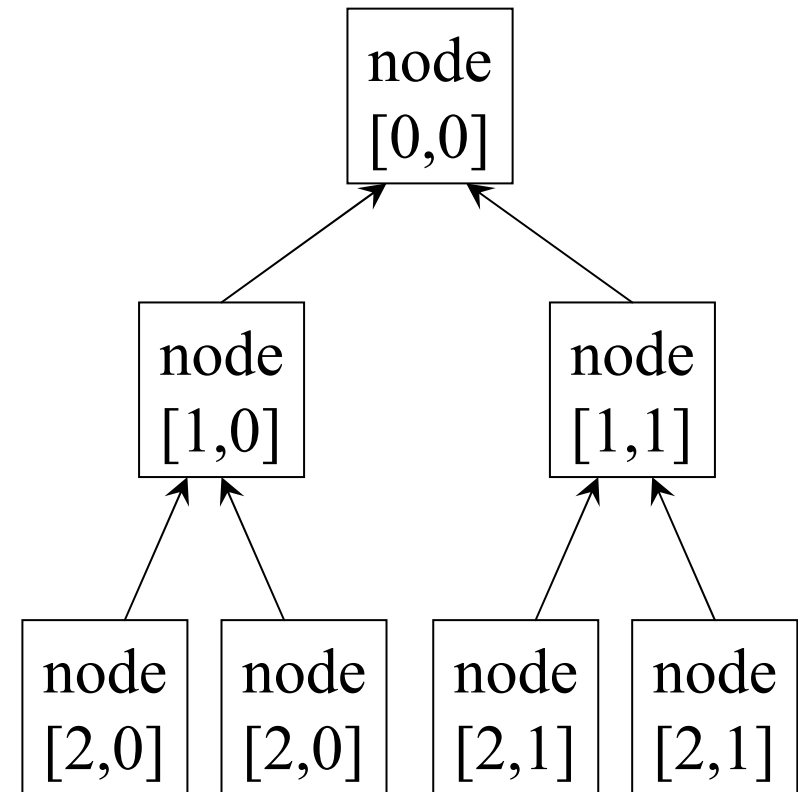
$$C_1 = 1, \dots, C_k = 1.$$

Parallelization on the Processor Level

Processor Hierarchy

The partial solutions of two (or more) processors can be transmitted to a next higher level and processed.

A complete tree of processors is appropriate – the processors' can work fully in parallel.



Reduction of Solution Sets

Alternative Approach

$$A \cap B \Leftrightarrow A \cap \overline{\overline{B}} \Leftrightarrow A \setminus \overline{B}$$

1. Take the solution set of the first disjunction.
2. Take the complement of the solution set of the second disjunction.
3. Use the difference of the first and the second set.

The difference of these sets will be smaller and smaller.

Reduction of Solution Sets

Example: $CF = D_1 \wedge D_2 = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) = 1$

$D_1 = 1$ with the solution set $\begin{pmatrix} 1 & - & - \\ 0 & 0 & - \\ 0 & 1 & 1 \end{pmatrix}$

$\bar{D}_2 = 1$ with the solution set $(0 \ 0 \ 0)$

Difference: $D_1 \setminus \bar{D}_2 = \begin{pmatrix} 1 & - & - \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & - & - \\ 0 & - & 1 \end{pmatrix}$

Reduction of Solution Sets

Parallel approach

The solution set S existing so far can be split into **orthogonal** subsets S_1 and S_2 .

Further processing can be submitted to two (or more) processors:

$$(S_1 \cup S_2) \setminus M = (S_1 \setminus M) \cup (S_2 \setminus M).$$



simple connection

Clause Ordering

Small intermediate sets can be achieved considering four situations:

1. Three values meet three variables.
2. Two values meet three variables.
3. One value meets three variables.
4. No value meets three variables.

Remember: the clauses always have three variables.

Clause Ordering

Case 1: No Expansion

values	clause	intersection
000	$\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 = (000)$	no orthogonality – one solution (000)
000	$x_1 \vee \bar{x}_2 \vee \bar{x}_3 = (100)$	one orthogonality – one solution (000)
000	$x_1 \vee x_2 \vee \bar{x}_3 = (110)$	two orthogonalities – one solution (000)
000	$x_1 \vee x_2 \vee x_3 = (111)$	three orthogonalities – unsatisfiable – delete vector

The clause can be deleted after this step.

Clause Ordering

Case 2: Unit Propagation

values	clause	solution
000 -	$\bar{x}_2 \vee \bar{x}_3 \vee x_4 = (-001)$	no orthogonality, solution (000 -)
000 -	$x_2 \vee \bar{x}_3 \vee x_4 = (-101)$	one orthogonality, solution (000 -)
000 -	$x_2 \vee x_3 \vee x_4 = (-111)$	two orthogonalities, solution (000 1)

The clause can be deleted after this step.

Clause Ordering

Case 3: Recursive Learning

values	clause	solution
000 - -	$\bar{x}_3 \vee x_4 \vee x_5 = (- - 011)$	no orthogonality – solution (000 - -)
000 - -	$x_3 \vee x_4 \vee x_5 = (- - 111)$	one orthogonality – solutions (000 1-) and (000 01)

The clause can be deleted after this step.

Clause Ordering

Case 4: Linear Expansion

No value meets the three variables:

The vector will be replaced by three vectors
with one, two or three components set.

This is very recommendable at the beginning of
the solution process.

values	clause	solutions
000 - - -	$x_4 \vee x_5 \vee x_6 = (- - - 1 1 1)$	(000 1 - -) (000 0 1 -) (000 0 0 1)

Clause Ordering

Equivalence Reasoning

$$(x_1 \vee \bar{x}_2 \vee x_3) (\bar{x}_1 \vee x_2 \vee x_4)$$

Solution vectors

$$(1 \ 1 \ - \ -), (0 \ 0 \ - \ -), (1 \ 0 \ - \ 1), (0 \ 1 \ 1 \ -)$$

Clause Ordering

Exclusive-or Reasoning

$$(x_1 \vee x_2 \vee x_3) (\bar{x}_1 \vee \bar{x}_2 \vee x_4)$$

Solution vectors

$$(1\ 0\ -\ -), (0\ 1\ -\ -), (1\ 1\ -\ 1), (0\ 0\ 1\ -)$$

Clause Ordering

Complement Reasoning

$$(x_1 \vee x_2 \vee x_3)(\bar{x}_1 \vee x_4 \vee x_5)$$

solution vectors

$$(1 \ - \ - \ 1-), (1 \ - \ - \ 0 \ 1), (0 \ 1 \ - \ - \ -), (0 \ 0 \ 1 \ - \ -)$$

Clause Ordering

The same variable in two clauses

$$\begin{array}{ccccccc} & & & x & \Rightarrow & x & \\ & & & & & & 1 \\ 1 & & & 1 & & & 0 \\ 0 & \setminus & & 0 & \Rightarrow & & 0 \\ 0 & & & 0 & & & 0 \\ & & & & & & 0 \end{array}$$

Clause Ordering

The same variable complemented in
the second clause

		x	\Rightarrow	\bar{x}	
1		0			1
0	\	1	\Rightarrow		1
0		1			0
					0

Clause Ordering

Ordering of the variables

The ordering of the variables according to their frequency defines the optimal sequence for the consideration of clauses:

1. Lexicographic Order.
2. Explicit consideration of the frequency of variables.

Further experiments on large multiprocessor systems are necessary!

Experimental Results

Basic Parallel Approach

A0: $S \leftarrow \text{CPL}(\text{NDM}(\text{CNF}(f)))$



Benchmark	number of		time in seconds				
	variables	clauses	A0	A111	A112	A1s	A1p
uf50-218-01	50	218	134	82	27	109	82

A1r: 6168 sec !



A1: $S \leftarrow \text{ISC}(\text{CPL}(\text{NDM}(\text{CNF}(f_1))), \text{CPL}(\text{NDM}(\text{CNF}(f_2))))$



Experimental Results

Influence of Ordering

Benchmark: uf50-218-01

operation	number of rows			time in seconds				
	A111	A112	A1r	A111	A112	A1r	A1s	A1p
sort()			218			0		
NDM 1	117			0				
CPL 1	15128			0				
NDM 2		101			0			
CPL 2		363220			9			
ISC			10			37		
all together			10	0	9	37	46	46

Experimental Results

Influence of the Algorithm

Benchmark: uf50-218-01

operation	number of rows			time in seconds				
	A2l1	A2l2	A2r	A2l1	A2l2	A2r	A2s	A2p
sort()			218			0.05		
DIF 1	10			0.03				
DIF 2		0			0.01			
CON			10			0.00		
all together			10	0.03	0.01	0.05	0.09	0.08

A2: $S1_1 \leftarrow \text{DIF}(S1_1, \text{NDM}(C_2, \dots, C_n))$
 $S2_1 \leftarrow \text{DIF}(S2_1, \text{NDM}(C_2, \dots, C_n))$
 $S \leftarrow \text{CON}(S1_1, S2_1)$

Experimental Results

Influence of the Rearrangement

Benchmark	number of		time in seconds			
	variables	clauses	A3	A3 A2	A4	A4 A2
uf75-325-01	75	325	0.07	4569	0.09	26

A3: Rearrange Order of Clauses by Influence of Variables

A4: Rearrange Order of Clauses by Number of Used Variables

Conclusions

- There are several approaches to split a SAT – problem into subproblems which can be solved in parallel on several processors:
 1. splitting of the set of clauses,
 2. splitting of the solution space.
- The algorithm of the second approach is more favorable.
- The order of the clause influences the solution efforts strongly.

Conclusions

- The algorithm can be extended to any kind of SAT-problems (not only 3-SAT). Only the coding of the initial solution sets must change.
- The algorithm can run in parallel on as many processors as available. Any hierarchy of processors is possible.
- This will allow the solution of problems of “any practical size”.

Note: Other developments in AI show the same tendency.