Cumulating Search in a Distributed Computing Environment

Hantao Zhang   Maria Paola Bonacina
Dept. of Computer Science
University of Iowa

Outline

- Introduction
- Satisfiability:
  Davis - Putnam
  SATO
- Parallel satisfiability:
  PSATO
- Experiments
- Problems on quasigroups
- Discussion

Satisfiability

S: set of propositional clauses

- Davis Putnam algorithm 1960

- NP - completeness , Cook 1971

Satisfiability

Recent work:

- Sequential SAT:

  [Zhang 1993] SATO

  [Zhang, Stickel 1994]

  [McCune 1994] ANL-DP

  ...

- Parallel / distributed SAT:

  [Böhm, Speckenmeyer 1994]

  [Zhang 1994] PSATO

  ...

Motivation

- SAT

- Constraint satisfaction problems

  on finite domains $\Longrightarrow$ SAT problems

- Model generation

  [Slaney 1992] FINDER

- Algebraic problems

  Quasigroups problems

  [J. Zhang 1990]

  [Fujita, Slaney, Bennett 1993]

...

Motivation

Problems in distributed deduction:

- Partition the search space among parallel processes
- Combine the work of parallel processes
- Cumulate the results of searches performed in separate intervals of time
- Scalability
- Fault-tolerance

PSATO: a parallel SAT prover

- concurrent asynchronous SAT processes
- one process on each node of a network of workstations
- partition search space
  no overlap
- cumulate search by concurrent processes
- cumulate search over time
- highly scalable
- fault-tolerant

The Davis-Putnam algorithm

- case splitting

$$\begin{tikzcd}[column sep=small] & S \arrow[dl] \arrow[dr] & \\ S[L \leftarrow \text{true}] & & S[L \leftarrow \text{false}] \end{tikzcd}$$

- unit clause rule

$$\begin{tikzcd} S \cup \{L\} \arrow[d] \\ S[L \leftarrow \text{true}] \end{tikzcd}$$

- pure literal rule

$$\begin{tikzcd} S \arrow[d, "L \text{ is pure in } S"] \\ S[L \leftarrow \text{true}] \end{tikzcd}$$

SATO

- case splitting

$$
\begin{tikzcd}[column sep=small]
 & S \arrow[dl] \arrow[dr] & \\
S \cup \{L\} & & S \cup \{\neg L\}
\end{tikzcd}
$$

- unit clause rule

$$
\left.\begin{array}{l}
\cdot\ \text{unit subsumption} \\
\cdot\ \text{unit resolution}
\end{array}\right\}
\begin{array}{l}
\text{unit} \\
\text{propagation}
\end{array}
$$

SATO

- Trie data structure (discrimination trees) for representing clauses.
- Trie-based sublinear algorithm for unit-propagation [Zhang, Stickel 1994]
- Heuristic for splitting: choose a literal in one of the shortest positive clauses.

Tries im SATO

Trie for L :

$$\langle L, T_L, T_{\neg L}, T_0 \rangle$$

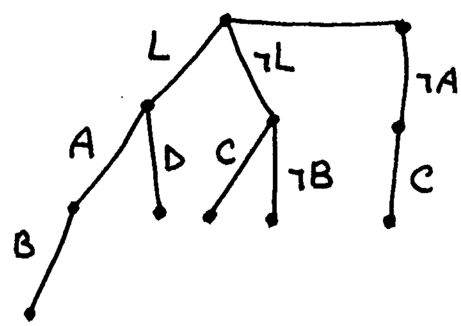
$L \vee A \vee B$

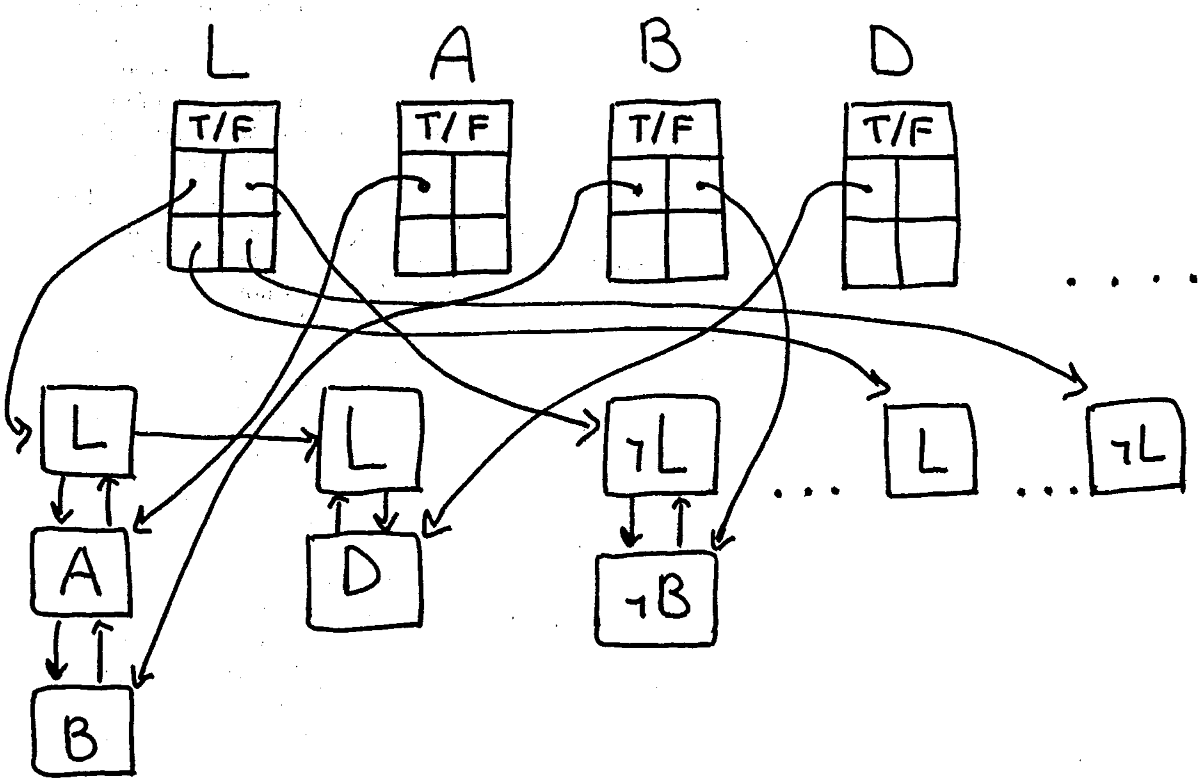
$L \vee D$

$\neg L \vee \neg B$

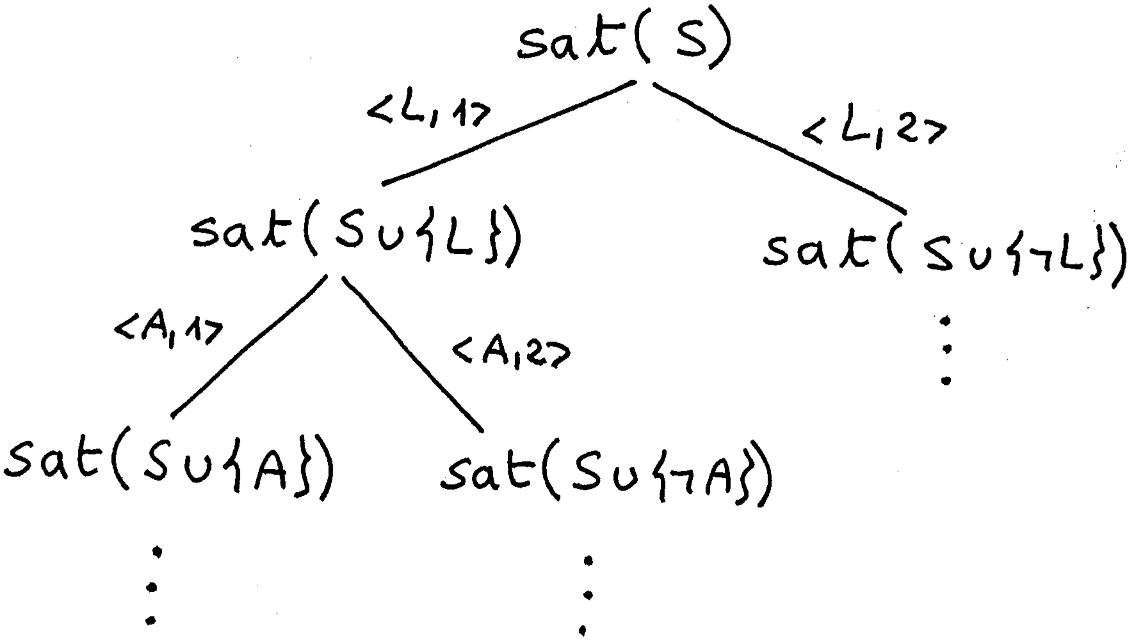
$\neg A \vee C$

$\neg L \vee C$





Cumulating search over time



guiding path : $(\langle L,1\rangle, \langle A,2\rangle, \ldots\ldots)$

"open" "closed"

sat(S, $\emptyset$)

halt at node x

save guiding path P from root to x

sat(S, P)

PSATO

Master / Slave organization.

The master:

- partition the search space among the slaves,
- broadcast
    - choice of splitting rule,
    - allotted time,
    - halting message ....
- receive reports.

Each slave executes the Davis-Putnam algorithm (SATO).

Partition the search space

The master assigns to each slave:

- the set of input clauses,
- a different guiding path.

Different guiding paths lead the search to disjoint portions of the search space.

$$P = (\langle L_1, 2\rangle \ldots \langle \underline{L_i, 1}\rangle \ldots \langle L_k, \delta_k\rangle)$$

is splitted into

$$P_1 = (\langle L_1, 2\rangle \ldots \langle \underline{\neg L_i, 2}\rangle)$$

$$P_2 = (\langle L_1, 2\rangle \ldots \langle \underline{L_i, 2}\rangle \ldots \langle L_k, \delta_k\rangle)$$

PSATO

The master handles a list of guiding paths to be assigned.

Each slave may report:

- true
- false
- out of time / memory with guiding path computed so far

$\exists$ guiding path returning true: satisfiable.

$\forall$ guiding paths return false: unsatisfiable.

Experiments with PSATO on random 3-SAT unsatisfiable problems:

#V	#P	Wall Clock	Total Time	Speed-up	Over-head
100	1	22.2	22.2	–	–
	5	7.9	24.4	2.81	0.10
	20	3.6	26.0	6.17	0.17
150	1	1082.5	1082.5	–	–
	5	237.9	1169.1	4.55	0.08
	20	60.7	1212.4	17.83	0.12
200	1	53346.7	53346.7	–	–
	5	10777.0	54947.1	4.95	0.05
	20	2899.3	58793.4	18.40	0.07

#V : number of propositional variables

#P : number of processors

Quasigroups

Groupoid : $\langle S, * \rangle$

Cancellative :

$$x * u = y \ , \ x * w = y \implies u = w$$

$$u * x = y \ , \ w * x = y \implies u = w$$

$$x * y = u \ , \ x * y = w \implies u = w$$

Finite : $S = \{0, \dots, v-1\}$

$|S| = v$ order

$$(x * y = 0) \vee \dots \vee (x * y = v-1)$$

$$(x * 0 = y) \vee \dots \vee (x * (v-1) = y)$$

$$(0 * x = y) \vee \dots \vee ((v-1) * x = y)$$

Quasigroup problems

[Fujita, Slaney, Bennett 1993]

Name	Constraint
QG1	$x * y = u, z * w = u, v * y = x,$ $v * w = z \Rightarrow x = z, y = w$
QG2	$x * y = u, z * w = u, y * v = x,$ $w * v = z \Rightarrow x = z, y = w$
QG3	$(x * y) * (y * x) = x$
QG4	$(x * y) * (y * x) = y$
QG5	$((x * y) * x) * x = y$
QG6	$(x * y) * y = x * (x * y)$
QG7	$((x * y) * x) * y = x$

QGi.v : basic axioms,

i-th constraint,

order v ($S = \{0, \ldots, v-1\}$),

$x * x = x$.

Quasigroup problems

Does there exist a quasigroup satisfying a specification QGi.v?

Replace variables in QGi.v by values in S.

Replace $x * y = z$ by $P_{x,y,z}$.

Satisfiability problem.

$O(v^k)$ clauses if QGi.v contains k variables.

Quasigroup problems

solved first by machine

Y : satisfiable

N : unsatisfiable

O : open

v :	9	10	11	12	13	14	15	16	17
QG1				O					
QG2		O		Y		Y	Y		
QG3				Y					
QG4				Y					
QG5	N	N	Y	N	N	N	N	N	O
QG6		N	N	N		N	N		Y
QG7		N	N	N		N	N		

Other open problems:

QG5.18, QG6.20, QG7.33.

Performance of PSATO on hard unsatisfiable problems

Prob.	#P	Workdays	P-Measure
QG5.14(*)	20	35	11
QG6.15	20	8	8
QG7.15	20	5	6
QG5.16	20	4	5
QG6.17	8	2	–

#P : number of processors

workday = 8 hours

(*) : non-idempotent

P-Measure : number of open pairs in guiding path after one workday.

If p-measure = n, then $O(2^n)$ workdays (empirical observation).

Discussion

- PSATO: a distributed prover for propositional satisfiability.
- Partition the search space effectively.
- Accumulate search over time.
- Use non-dedicated general-purpose networks of workstations effectively.
- Achieve high scalability and fault-tolerance by the master / slave organization.
- Experiments: quasigroup problems.