PaSAT – Parallel SAT-Checking with Lemma Exchange

Carsten Sinz, Wolfgang Blochinger and Wolfgang Küchlin
Symbolic Computation Group
University of Tübingen, Germany

http://www-sr.informatik.uni-tuebingen.de
Introduction

Goal:
Speed up DP algorithm for encodings of real-world combinatorial problems

Instruments:
- Parallelization (dyn. search-space partitioning)
- Learning (lemma generation)

PaSAT:
Combination of both acceleration methods
Basic Davis-Putnam Algorithm

boolean DP(Clause Set S)
{
    while (S contains a unit clause {L}) {  // unit prop.
        delete from S all clauses containing L;  // u. subs.
        delete ¬L from all clauses in S;      // u. res.
    }
    if (∅ ∈ S) return false;
    if (S = ∅) return true;
    choose a literal L occurring in S;
    if (DP(S  # {{L}}) return true;
    else return DP(S  # {{¬L}});
}

Recent Improvements on DP-based Solvers

- Efficient implementation of UP and backtracking
  - Head/tail lists [H.Zhang, Stickel ´96]
  - Watched literals [Moskewicz et al. ´01]
- Improved variable selection strategies
  - Look-ahead [Li ´97], “decaying sum” [Moskewicz et al. ´01], combinations [H.Zhang ´97]
- Randomization, search restarts
  - Overcome limitations of heuristics (influenced by local search algorithms) [Gomes/Selman/Kautz ´98]
- Clause set „compression“
  - Deletion of subsumed clauses [L.Zhang, Malik ´01]
- Lemma generation [Marques-Silva, Sakallah ´96]
Parallelization Method

- Guiding path [H. Zhang et al., ´96] describes state of search, e.g.
  \[((x, B), (y, N), (z, B), (u, B))\]

- Partitioning of search-space at each (_,B) entry possible, e.g.
  \[((x, N), (\overline{y}, N), (z, B), (u, B))\]
  \[((\overline{x}, N))\]
Lemma Generation [Marques-Silva, Sakallah]

\[(\vec{u}, f, m)\]
\[(\vec{u}, \bar{m}, h)\]
\[(\bar{f}, \bar{g}, \bar{h})\]
\[(y, f)\]
\[(\bar{f}, g)\]

\[
\begin{align*}
    u &= 1 \\
    f &= \bar{u} \\
    g &= \bar{m} \\
    h &= \bar{f} \\
    x &= 1 \\
    d &= 0 \\
    e &= 0 \\
    f &= \bar{g} \\
    h &= \bar{f} \\
    y &= 0 \\
    k &= 0 \\
    z &= 1 \\
\end{align*}
\]

Conflict-induced clause: \[(y, \bar{u})\]
Challenges of Combining Both Ideas

- Cooperation needed to make full use of lemmas
- Acceleration by lemma generation may limit speed-ups attainable by parallelization:

![Diagram showing sequential and parallel runtime comparison](image-url)
Analyzing the Learning Effect

- Fraction \( f(t) \) of traversed search space at time \( t \) described by current guiding path \( P(t) \):
  \[
P(t) = ((L_1, b_1), \ldots, (L_d, b_d))
  \]
  \[
f(t) = \sum_{i=1}^{d} w(b_i) \cdot 2^{-i} \text{ where } w(N) = 1, w(B) = 0
  \]

- Effect \( e(x) \) of learning when fraction \( x \) of search is completed:
  \[
e(x) = \frac{df_1}{dt} \circ f_1^{-}(x), \quad x \in [0,1]
  \]
  \[
  ((x, B), (y, N), (z, B), (u, B))
  \]
Analyzing the Learning Effect (cont’d)

\[ t_{\text{seq}, n} = 3606.13 \quad t_{\text{seq}, l} = 41.10 \]
\[ t_{\text{par}, n} = 996.47 \quad t_{\text{par}, l} = 55.02 \]
\[ s_{\text{par/seq}, n} = 3.62 \quad s_{\text{par/seq}, l} = 0.75 \]

(parallel times on 4 processors)
**Parallelization Platform**

DOTS (Distributed Object-Oriented Threads System):

- Fork/join-(threads-)paradigm for both parallel and distributed computing
- C++
- Supported Systems:
  - Windows 98/NT/2000
  - Solaris, IRIX, AIX, FreeBSD, Linux
  - IBM zSeries z/OS
- Different load-distribution schemes, e.g.
  - work-sharing
  - work-stealing
### Experimental Results

<table>
<thead>
<tr>
<th>problem instance</th>
<th>$t_{\text{Seq}}$</th>
<th>$t_{\text{Par}}$</th>
<th>$t_{\text{PaSAT}}$</th>
<th>$s_{\text{Par/Seq}}$</th>
<th>$s_{\text{Par/PaSAT}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>longmult7</td>
<td>73.78</td>
<td>22.24</td>
<td>15.49</td>
<td>3.32</td>
<td>4.76</td>
</tr>
<tr>
<td>longmult8</td>
<td>176.91</td>
<td>51.54</td>
<td>42.76</td>
<td>3.43</td>
<td>4.14</td>
</tr>
<tr>
<td>longmult9</td>
<td>291.00</td>
<td>79.51</td>
<td>77.78</td>
<td>3.66</td>
<td>3.74</td>
</tr>
<tr>
<td>longmult10</td>
<td>414.49</td>
<td>113.92</td>
<td>138.71</td>
<td>3.64</td>
<td>2.99</td>
</tr>
<tr>
<td>longmult11</td>
<td>541.14</td>
<td>145.81</td>
<td>179.28</td>
<td>3.71</td>
<td>3.02</td>
</tr>
<tr>
<td>longmult12</td>
<td>646.43</td>
<td>163.66</td>
<td>162.78</td>
<td>3.95</td>
<td>3.97</td>
</tr>
<tr>
<td>longmult13</td>
<td>809.67</td>
<td>202.92</td>
<td>227.93</td>
<td>3.99</td>
<td>3.55</td>
</tr>
<tr>
<td>longmult14</td>
<td>929.21</td>
<td>242.13</td>
<td>248.13</td>
<td>3.84</td>
<td>3.74</td>
</tr>
</tbody>
</table>

Sun ES450, 4 UltraSparcII processors @ 400MHz, 1GB
**Conclusion**

- **Combination of learning/parallelization:**
  - Cooperation by exchanging suitable lemmas
  - Makes use of networks of multiprocessor workstations
  - Delivers good results on real-world SAT-instances

- **Preliminary analysis of learning behavior:**
  - Allows graphical presentation of learning effect over time

- **Main result:**
  Parallelization and learning perfectly complement each other.