Résolution parallèle de SAT : mieux collaborer pour aller plus loin

Gilles Audemard, Benoît Hoessen, Saïd Jabbour, Jean-Marie Lagniez, Cédric Piette

Université Lille-Nord de France
CRIL - CNRS UMR 8188
Artois, F-62307 Lens
{audemard, hoessen, jabbour, piette}@cril.fr

Institute for Formal Models and Verification
Johannes Kepler University,
AT-4040 Linz, Austria
{jean-marie.lagniez}@jku.at

May 22, 2012

This work has been supported by CNRS and OSEO, under the ISI project “Pajero”.
SAT framework
SAT framework

- Lots of solvers are of type CDCL
SAT framework

- Lots of solvers are of type CDCL
- CDCL creates one new clause at each conflict
SAT framework

- Lots of solvers are of type CDCL
- CDCL creates one new clause at each conflict
- Restarts are quite frequent
SAT framework

- Lots of solvers are of type **CDCL**

- **CDCL** creates one new clause at each conflict

- Restarts are quite frequent

- **lbd** provide a qualitative measure about clauses
Freezing clauses

Classical clause management

- $\mathcal{A}$: Set of active clauses
- $\mathcal{D}$: Set of deleted clauses
Freezing clauses

Freezing clause management

- $\mathcal{A}$: Set of active clauses
- $\mathcal{D}$: Set of deleted clauses
- $\mathcal{F}$: Set of frozen clauses

Diagram:

- $\mathcal{A}$ to $\mathcal{F}$ with label "not psm-cond"
- $\mathcal{F}$ to $\mathcal{D}$ with label "not activated"
- $\mathcal{D}$ to $\mathcal{A}$ with label "not used"
- $\mathcal{A}$ to $\mathcal{D}$ with label "psm-cond"
How?

Two main approaches
How?

Two main approaches

Divide and conquer
How?

Two main approaches

Divide and conquer

\[
\begin{array}{c}
\Sigma \\
a=\text{true} & a=\text{false} \\
\Sigma_1 & \Sigma_2
\end{array}
\]
How?

Two main approaches

Divide and conquer

Portofolio

\[
\Sigma \\
\downarrow \\
a=true \quad a=false \\
\downarrow \\
\Sigma_1 \quad \Sigma_2
\]
How?

Two main approaches

Divide and conquer

\[ \Sigma \quad a=true \quad a=false \quad \Sigma \]

\[ \Sigma_1 \quad \Sigma_2 \]

Portfolio

\[ \Sigma \quad \Sigma \quad \Sigma \]

\[ m_1 \quad m_2 \quad m_3 \]
How?

Two main approaches

Divide and conquer

\[ \Sigma \]

\[ \Sigma_1 \quad \Sigma_2 \]

\[ a=true \quad a=false \]

Portofolio

\[ \Sigma \]

\[ \Sigma_1 \quad \Sigma_2 \quad \Sigma_3 \]

\[ m_1 \quad m_2 \quad m_3 \]

We will use the portofolio methodology
SAT competition 2011

ppfolio
SAT competition 2011

ppfolio

- run completely different state-of-the-art solvers in parallel
SAT competition 2011

ppfolio

- run completely different state-of-the-art solvers in parallel
- trusted the competition (16 medals)
SAT competition 2011

*ppfolio*

- run completely different state-of-the-art solvers in parallel
- trusted the competition (16 medals)
- the solvers do not communicate!
SAT competition 2011

*ppfolio*

- run completely different state-of-the-art solvers in parallel
- trusted the competition (16 medals)
- the solvers do not communicate!

Work need to be done on communication
Good communication?
Good communication?

Good communication
To achieve good communication, we need to **maximize** the exchange of **useful** information, and **minimize** the **useless** information.
Communication in portofolio

Good communication
Communication in portfolio

Good communication

- Information = clauses
Communication in portofolio

Good communication

- Information = clauses

- What is a useful clause?
Communication in portofolio

Good communication

- Information = clauses
- What is a useful clause?
- A useless clause is never used in propagation
New ratio
New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread $t$
New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread $t$
- $\text{used}(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread $t$
New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread $t$
- $\text{used}(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread $t$
- $\text{unused}(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread $t$ without being used
New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread $t$
- $\text{used}(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread $t$
- $\text{unused}(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread $t$ without being used
- $\#\mathcal{I}_t - \text{used}(\mathcal{I}_t, t) - \text{unused}(\mathcal{I}_t, t)$ the number of clauses in the database that are neither used, nor deleted by thread $t$
New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread $t$
- $used(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread $t$
- $unused(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread $t$ without being used
- $\#\mathcal{I}_t - used(\mathcal{I}_t, t) - unused(\mathcal{I}_t, t)$ the number of clauses in the database that are neither used, nor deleted by thread $t$

Usage ratio

$$\frac{\sum_{t=0}^{n} used(\mathcal{I}_t, t)}{\sum_{t=0}^{n} \#\mathcal{I}_t}$$
New ratio

- $\mathcal{I}_t$ the set of imported clauses by thread $t$
- $\text{used}(\mathcal{I}_t, t)$ the number of clauses imported and used in propagation by thread $t$
- $\text{unused}(\mathcal{I}_t, t)$ the number of clauses that were deleted by thread $t$ without being used
- $\#\mathcal{I}_t - \text{used}(\mathcal{I}_t, t) - \text{unused}(\mathcal{I}_t, t)$ the number of clauses in the database that are neither used, nor deleted by thread $t$

**Usage ratio**

$$\frac{\sum_{t=0}^{n} \text{used}(\mathcal{I}_t, t)}{\sum_{t=0}^{n} \#\mathcal{I}_t}$$

**Non-usage ratio**

$$\frac{\sum_{t=0}^{n} \text{unused}(\mathcal{I}_t, t)}{\sum_{t=0}^{n} \#\mathcal{I}_t}$$
Classic manysat

Usage ratio

Non-usage ratio

0 0.2 0.4 0.6 0.8 1

0 0.2 0.4 0.6 0.8 1

SAT
UNSAT
UNKNOWN
Classic manysat

- Ratio is good
Classic manysat

- Ratio is good
- A lot of imported clauses are not used but kept in memory
Challenges

Problems we must face
Challenges

Problems we must face

- Importation of duplicate information
Challenges

Problems we must face

- Importation of duplicate information
- Imported clauses can be useless for the current search subspace
Challenges

Problems we must face

- Importation of duplicate information
- Imported clauses can be useless for the current search subspace
- Higher number of learnt clauses
Introducing PeneLoPe

We want to design a solver based on ManySat 2.0 able to:
Introducing PeneLoPe

We want to design a solver based on ManySat2.0 able to:

- handle all the learnt clauses
Introducing PeneLoPe

We want to design a solver based on ManySat2.0 able to:

- handle all the learnt clauses
- communicate efficiently
Introducing PeneLoPe

We want to design a solver based on ManySat 2.0 able to:

- handle all the learnt clauses
- communicate efficiently
- use every processor on the host
Freeze in parallel
Freeze in parallel
Freeze in parallel

- Each thread has its own sets
Import policy
Import policy

- Freeze-all
Import policy

- Freeze-all
- Freeze
Import policy

- Freeze-all
- Freeze
- No freeze
Affecting the ratio

We could change the ratio by:
Affecting the ratio

We could change the ratio by:

- Restart strategy
Affecting the ratio

We could change the ratio by:

- Restart strategy
  - Luby technique
Affecting the ratio

We could change the ratio by:

- Restart strategy
  - Luby technique
  - \( lbd \) restarts
Affecting the ratio

We could change the ratio by:

- Restart strategy
  - Luby technique
  - $lbd$ restarts

- Choosing what is exported
Affecting the ratio

We could change the ratio by:

- Restart strategy
  - Luby technique
  - lbd restarts

- Choosing what is exported
  - Export every generated clauses
Affecting the ratio

We could change the ratio by:

- Restart strategy
  - Luby technique
  - $lbd$ restarts

- Choosing what is exported
  - Export every generated clauses
  - Export clauses of size $\leq s$
Affecting the ratio

We could change the ratio by:

- Restart strategy
  - Luby technique
  - $lb d$ restarts

- Choosing what is exported
  - Export every generated clauses
  - Export clauses of size $\leq s$
  - Export clauses with literal block distance $\leq l$
Comparison of combinations
Comparison of combinations

- policies have effects on each other
Comparison of combinations

- policies have effects on each other

- winning policy on our experiments:
  - export: \textit{lbd} based
  - import: \textit{no freeze}
  - restarts: \textit{lbd} based.
Comparison of combinations

- policies have effects on each other
- winning policy on our experiments:
  - export: lbd based
  - import: no freeze
  - restarts: lbd based.

<table>
<thead>
<tr>
<th></th>
<th>#SAT</th>
<th>#UNSAT</th>
<th>#SAT + #UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manysat</td>
<td>95</td>
<td>93</td>
<td>188</td>
</tr>
<tr>
<td>PeneLoPe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Comparison of combinations

- policies have effects on each other

- winning policy on our experiments:
  - export: *lbd* based
  - import: *no freeze*
  - restarts: *lbd* based.

<table>
<thead>
<tr>
<th></th>
<th>#SAT</th>
<th>#UNSAT</th>
<th>#SAT + #UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manysat</td>
<td>95</td>
<td>93</td>
<td>188</td>
</tr>
<tr>
<td>PeneLoPe</td>
<td>94</td>
<td>111</td>
<td>205</td>
</tr>
</tbody>
</table>
Comparison with other solvers

![Graph comparing WC time (seconds) vs nb instances for different solvers including PeneLoPe freeze, CryptoMiniSAT, Plingeling, PPfolio, and ManySAT.](image)
Conclusion
Conclusion

- We need to pay attention to clause exchange technique
Conclusion

- We need to pay attention to clause exchange technique
- The prototype is highly competitive
Conclusion

- We need to pay attention to clause exchange technique.
- The prototype is highly competitive.
- We can expend the orthogonality of the threads by using different techniques for each thread.
Thank you for your attention
Questions?
Some comparisons

<table>
<thead>
<tr>
<th>psm used</th>
<th>export strategy</th>
<th>restart strategy</th>
<th>import strategy</th>
<th>#SAT</th>
<th>#UNSAT</th>
<th>#SAT + #UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>lbd limit</td>
<td>lbd</td>
<td>no freeze</td>
<td>94</td>
<td>111</td>
<td>205</td>
</tr>
<tr>
<td>✓</td>
<td>lbd limit</td>
<td>lbd</td>
<td>freeze</td>
<td>89</td>
<td>113</td>
<td>202</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>lbd</td>
<td>freeze</td>
<td>93</td>
<td>107</td>
<td>200</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>lbd</td>
<td>no freeze</td>
<td>89</td>
<td>107</td>
<td>196</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>luby</td>
<td>no freeze</td>
<td>97</td>
<td>98</td>
<td>195</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>luby</td>
<td>freeze all</td>
<td>89</td>
<td>102</td>
<td>191</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>luby</td>
<td>freeze all</td>
<td>96</td>
<td>92</td>
<td>188</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>luby</td>
<td>freeze</td>
<td>92</td>
<td>96</td>
<td>188</td>
</tr>
<tr>
<td>✓</td>
<td>unlimited</td>
<td>lbd</td>
<td>freeze</td>
<td>86</td>
<td>102</td>
<td>188</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>luby</td>
<td>freeze</td>
<td>92</td>
<td>96</td>
<td>188</td>
</tr>
<tr>
<td>✓</td>
<td>unlimited</td>
<td>luby</td>
<td>freeze</td>
<td>91</td>
<td>97</td>
<td>188</td>
</tr>
<tr>
<td>manysat</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>95</td>
<td>93</td>
<td>188</td>
</tr>
<tr>
<td>✓</td>
<td>lbd limit</td>
<td>luby</td>
<td>no freeze</td>
<td>90</td>
<td>94</td>
<td>184</td>
</tr>
<tr>
<td>✓</td>
<td>unlimited</td>
<td>luby</td>
<td>freeze</td>
<td>91</td>
<td>92</td>
<td>183</td>
</tr>
<tr>
<td>✓</td>
<td>size limit</td>
<td>luby</td>
<td>no freeze</td>
<td>92</td>
<td>90</td>
<td>182</td>
</tr>
<tr>
<td>✓</td>
<td>unlimited</td>
<td>luby</td>
<td>no freeze</td>
<td>89</td>
<td>88</td>
<td>177</td>
</tr>
<tr>
<td>✓</td>
<td>size = 1</td>
<td>lbd</td>
<td>freeze</td>
<td>89</td>
<td>88</td>
<td>177</td>
</tr>
</tbody>
</table>
8 cores details

<table>
<thead>
<tr>
<th>Solver</th>
<th>#SAT</th>
<th>#UNSAT</th>
<th>#SAT+#UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeneLoPe <em>freeze</em></td>
<td>97</td>
<td>119</td>
<td>216</td>
</tr>
<tr>
<td>PeneLoPe <em>no freeze</em></td>
<td>96</td>
<td>119</td>
<td>215</td>
</tr>
<tr>
<td>plingeling</td>
<td>99</td>
<td>97</td>
<td>196</td>
</tr>
<tr>
<td>ppfolio</td>
<td>91</td>
<td>103</td>
<td>194</td>
</tr>
<tr>
<td>cryptominisat</td>
<td>89</td>
<td>104</td>
<td>193</td>
</tr>
<tr>
<td>ManySat</td>
<td>95</td>
<td>92</td>
<td>187</td>
</tr>
</tbody>
</table>
### 32 cores details

<table>
<thead>
<tr>
<th>Solver</th>
<th>#SAT</th>
<th>#UNSAT</th>
<th>#SAT+#UNSAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PeneLoPe freeze</td>
<td>104</td>
<td>127</td>
<td>231</td>
</tr>
<tr>
<td>PeneLoPe no freeze</td>
<td>99</td>
<td>131</td>
<td>230</td>
</tr>
<tr>
<td>ManySat</td>
<td>105</td>
<td>111</td>
<td>216</td>
</tr>
<tr>
<td>ppfolio</td>
<td>107</td>
<td>97</td>
<td>204</td>
</tr>
<tr>
<td>cryptominisat</td>
<td>96</td>
<td>105</td>
<td>201</td>
</tr>
<tr>
<td>plingeling</td>
<td>100</td>
<td>95</td>
<td>195</td>
</tr>
</tbody>
</table>
Scaling up to 32 cores

![Graph showing scaling performance of different SAT solvers.](image)
Literal block distance
Literal block distance

Definition

Given a clause $C$, and a partition of its literals into $n$ subsets according to the current assignment, s.t. literals are partitioned w.r.t their decision level. The lbd of $C$ is exactly $n$. 
Literal block distance

Definition
Given a clause $C$, and a partition of its literals into $n$ subsets according to the current assignment, s.t. literals are partitioned w.r.t their decision level. The $lbd$ of $C$ is exactly $n$.

$lbd$ restarts
$Avg_s$ is the average of $lbd$ of the clauses created since the start of the process. $Avg_{100}$ is the average of $lbd$ of the last 100 created clauses. Restarts when $Avg_{100} \times \alpha \geq Avg_s$, $\alpha = 0.7$