APT-PBO: improving the Software Dependency Problem using Pseudo-Boolean Optimization

FOSDEM 2011 - Brussels

Paulo Trezentos

February 6, 2011
Agenda

1. Introduction
   - Problem

2. APT-PBO
   - Architecture
   - Algorithm

3. Experimental Results
   - Unmet dependencies analysis
   - Meta-installers performance
   - Solutions assessment
   - Multicriteria

4. Conclusions
Agenda

1. Introduction
   - Problem

2. APT-PBO
   - Architecture
   - Algorithm

3. Experimental Results
   - Unmet dependencies analysis
   - Meta-installers performance
   - Solutions assessment
   - Multicriteria

4. Conclusions
Installing Linux packages (software components) is...

Introduction to the problem

- **Relevance:**
  - Installation process is critical for Operating Systems dissemination
Introduction to the problem

- **Relevance:**
  - Installation process is critical for Operating Systems dissemination

- **Problem:**
  - It is often reported that problems of broken dependencies between software packages in Linux

---

**Related work:**
- EDOS, MANCOOSI, Opium / Linspire, SuSE Libzypp, Eclipse P2 / SAT4J,...
Introduction to the problem

- **Relevance:**
  - Installation process is critical for Operating Systems dissemination

- **Problem:**
  - It is often reported that problems of broken dependencies between software packages in Linux

- **Solution:**
  - Use SAT to solve dependencies or, better, use PBO to choose between solutions
Introdução ao problema

● Relevância:
  ● Instalação process is critical for Operating Systems dissemination

● Problema:
  ● It is often reported that problems of broken dependencies between software packages in Linux

● Solução:
  ● Use SAT to solve dependencies or, better, use PBO to choose between solutions

● Trabalho relacionado:
  ● EDOS, MANCOOSI, Opium / Linspire, SuSE Libzypp, Eclipse P2 / SAT4J,...
Dependencies solving

- Install car
- 
- 
-
Dependencies solving

- Install car
- Need to install the dependencies / avoid conflicts
- NP-problem
Agenda

1. Introduction
   - Problem

2. APT-PBO
   - Architecture
   - Algorithm

3. Experimental Results
   - Unmet dependencies analysis
   - Meta-installer performance
   - Solutions assessment
   - Multicriteria

4. Conclusions
APT-PBO Architecture

- apt-pbo install car
- apt-get pboinstall car
- PBO solver
- process solution
- apt-get install solution

- pboinstall: encoding of the problem

[installed packages]
APT-PBO Architecture

- **pboinstall**: encoding of the problem
- **PBO solver**: external solver

```plaintext
apt-pbo install car
[installed packages]
```
APT-PBO Architecture

apt-pbo install car

- apt-get pboinstall car
- PBO solver
- process solution
- apt-get install solution

[installed packages]

- pboinstall: encoding of the problem
- PBO solver: external solver
- Parsing / processing: analyse the solution and iterate
APT-PBO Architecture

- `apt-pbo install car`
- `apt-get pboinstall car`
- `PBO solver`
- `process solution`
- `apt-get install solution`

- `pboinstall`: encoding of the problem
- `PBO solver`: external solver
- `Parsing / processing`: analyse the solution and iterate
- `apt-get install`: install packages

[installed packages]
APT-PBO Algorithm

Require: Package to install $p_1$, $Pol$
1: repeat
2: $(f, c) \leftarrow \text{call\_pboinstall}(p_1, P_c, Pol, R, PI)$
3: $S \leftarrow \text{call\_solver}(f, c)$
4: $P_c \leftarrow 0$
5: for all $p_i \in S$ such that $p_i = 0$ do
6: $P_c \leftarrow \text{check\_rdeps}(p_i)$
7: end for
8: for all $p_j \in S$ such that $p_j = 1$ do
9: $P_c \leftarrow \text{check\_rconfs}(p_j)$
10: end for
11: until $P_c = 0$
12: return $f, c$
Multicriteria

- If we try to satisfy different criteria, we have a MCDM (Multicriteria Decision Making problem)
Multicriteria

- If we try to satisfy different criteria, we have a MCDM (Multicriteria Decision Making problem)

- APT-PBO integrates different objective function as a multiobjective problem (MOP) and transforming into a single objective problem through *weighted sum scalarization*:

\[
\min \sum_{k=1}^{3} W_k \cdot f_k(P)
\]

and therefore:

\[
\min (W_r \cdot f_1(P) + W_p \cdot f_2(P) + W_v \cdot f_3(P))
\]
# Agenda

1. **Introduction**
   - Problem

2. **APT-PBO**
   - Architecture
   - Algorithm

3. **Experimental Results**
   - Unmet dependencies analysis
   - Meta-installers performance
   - Solutions assessment
   - Multicriteria

4. **Conclusions**
Unmet dependencies analysis

<table>
<thead>
<tr>
<th>Meta-installer</th>
<th>Possible Solutions</th>
<th>No Solutions</th>
<th>Wrong Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apt-get</td>
<td>591</td>
<td>123</td>
<td>0</td>
</tr>
<tr>
<td>Aptitude</td>
<td>713</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Smart</td>
<td>714</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Apt-pbo</td>
<td>714</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Meta-installer performance

<table>
<thead>
<tr>
<th>Meta-installer</th>
<th>Average Time</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apt-get</td>
<td>00:00.21</td>
<td>00:00.17</td>
</tr>
<tr>
<td>Aptitude</td>
<td>00:00.62</td>
<td>00:00.15</td>
</tr>
<tr>
<td>Smart</td>
<td>00:02.63</td>
<td>00:00.25</td>
</tr>
<tr>
<td>Apt-pbo</td>
<td>00:03.77</td>
<td>00:01.91</td>
</tr>
</tbody>
</table>
Solutions assessment

- Each package transaction can include installation, update, removal or downgrade

In 1,000 package transactions, how each meta-installer behaves?

<table>
<thead>
<tr>
<th>Tool</th>
<th>Installed</th>
<th>Updated</th>
<th>Removed</th>
<th>Down-graded</th>
</tr>
</thead>
<tbody>
<tr>
<td>apt-get</td>
<td>7,766</td>
<td>17</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>aptitude</td>
<td>8,423</td>
<td>18</td>
<td>161</td>
<td>464</td>
</tr>
<tr>
<td>smart</td>
<td>7,786</td>
<td>92</td>
<td>124</td>
<td>479</td>
</tr>
<tr>
<td>PBO freshness</td>
<td>6,808</td>
<td>20,449</td>
<td>25</td>
<td>567</td>
</tr>
<tr>
<td>PBO removal</td>
<td>7,767</td>
<td>11</td>
<td>162</td>
<td>443</td>
</tr>
<tr>
<td>PBO number</td>
<td>7,729</td>
<td>13</td>
<td>102</td>
<td>500</td>
</tr>
</tbody>
</table>
Solutions assessment

![Solutions assessment chart]

The chart compares different solutions in terms of their performance in handling various actions such as installation, updates, removal, and downgrades. The solutions assessed are Apt-get, Aptitude, Smart, PBO-Freshness, PBO-Removal, and PBO-Number.

- **Apt-get**: The highest percentage of total actions are installed.
- **Aptitude**: A significant portion is dedicated to updates.
- **Smart**: The most actions are downgrades.
- **PBO-Freshness**: A notable number of updates and removals.
- **PBO-Removal**: Mostly removals.
- **PBO-Number**: Predominantly installations.

The overall performance of each solution is evaluated in terms of their effectiveness in managing dependencies and ensuring system stability.
Multi-criteria

In APT-PBO, varying the weights, we can have different solutions.
Multi-criteria

- In APT-PBO, varying the weights, we can have different solutions.
- For example, varying freshness weight we can obtain different solutions for package at-spi:

<table>
<thead>
<tr>
<th></th>
<th>Aggressive</th>
<th>Conservative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install</td>
<td>9</td>
<td>37</td>
</tr>
<tr>
<td>Updates</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>Remove</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Downgrade</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Time</td>
<td>00:19.45</td>
<td>00:17.37</td>
</tr>
</tbody>
</table>
**ap-spi - conservative freshness**

- To avoid the removal of “file-roller” we perform the installation of 28 extra-packages.
ap-spi - agressive freshness

- With “just” one removal of “file-roller” with avoid the installation of 28 packages and the downgrade of 3.
Agenda

1. Introduction
   - Problem

2. APT-PBO
   - Architecture
   - Algorithm

3. Experimental Results
   - Unmet dependencies analysis
   - Meta-installers performance
   - Solutions assessment
   - Multicriteria

4. Conclusions
Conclusions

APT-PBO is a freely available tool (GPL) that can be tested, available for Debian, Mandriva and Caixa Magica.
Conclusions

- APT-PBO is a freely available tool (GPL) that can be tested, available for Debian, Mandriva and Caixa Magica.
- PBO encoding proved to be suitable for the problem and provides extra-flexibility compared with SAT tools.
Conclusions

- APT-PBO is a freely available tool (GPL) that can be tested, available for Debian, Mandriva and Caixa Magica
- PBO encoding proved to be suitable for the problem and provides extra-flexibility compared with SAT tools
- Multi-criteria is a desirable and achievable option
Conclusions

- APT-PBO is a freely available tool (GPL) that can be tested, available for Debian, Mandriva and Caixa Magica.
- PBO encoding proved to be suitable for the problem and provides extra-flexibility compared with SAT tools.
- Multi-criteria is a desirable and achievable option.
- Performance is an issue for large repositories and some combination of weights.
Future work

- Better tuning of PBO encoding and work with PBO solver researchers to improve performance issues
Future work

- Better tuning of PBO encoding and work with PBO solver researchers to improve performance issues
- Support new features like using APT-PBO for removals, etc.
Future work

- Better tuning of PBO encoding and work with PBO solver researchers to improve performance issues
- Support new features like using APT-PBO for removals, etc.
- Enhance interactive mode
More information


- http://aptpbo.caixamagica.pt
Thank you.

Questions.