Boolean Leximax Optimisation using Iterative SAT Solving

Miguel Cabral\textsuperscript{1}  
Mikoláš Janota\textsuperscript{2}  
Vasco Manquinho\textsuperscript{1}

SAT 2022

\textsuperscript{1} Universidade de Lisboa  
\textsuperscript{2} Czech Technical University in Prague
Package Upgradeability Problem

$ apt install libreoffice
$ apt install libreoffice
libreoffice requires:
  • libreoffice-writer
  • libreoffice-calc
Package Upgradeability Problem

$ apt install libreoffice
libreoffice requires:
  • libreoffice-writer
  • libreoffice-calc
libreoffice-calc requires:
  • ure
$ apt install libreoffice

libreoffice requires:
  • libreoffice-writer
  • libreoffice-calc

libreoffice-calc requires:
  • ure ← several versions
Package Upgradeability Problem

$ apt install libreoffice
libreoffice requires:
  • libreoffice-writer
  • libreoffice-calc
libreoffice-calc requires:
  • ure ← several versions
ure, 7645 requires:
  • debconf $\geq 1675$ or debconf-2.0
$ apt install libreoffice
libreoffice requires:
  • libreoffice-writer
  • libreoffice-calc
libreoffice-calc requires:
  • ure ← several versions
ure, 7645 requires:
  • debconf ≥ 1675 or debconf-2.0
ure, 7645 conflicts with:
  • cli-uno-bridge < 16229
# Package Upgradeability Problem

<table>
<thead>
<tr>
<th>System</th>
<th>Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>ubuntu</td>
<td>apt</td>
</tr>
<tr>
<td>fedora</td>
<td>dnf</td>
</tr>
<tr>
<td>python</td>
<td>pip</td>
</tr>
<tr>
<td>OCaml</td>
<td>opam</td>
</tr>
</tbody>
</table>
Package Upgradeability Problem

Package Upgradeability $\sim$ SAT
Package Upgradeability Problem

Package Upgradeability $\leadsto$ SAT

**Minimize:**
Package Upgradeability Problem

Package Upgradeability $\iff$ SAT

Minimize:

- Number of **removed** packages
- Number of not **up-to-date** packages
- ...


Package Upgradeability Problem

Package Upgradeability $\leadsto$ SAT

Minimize:

- Number of removed packages
- Number of not up-to-date packages
- ...

Multi-Objective MaxSAT

Hard clauses + Multiple sets of Soft clauses
Multiple Objective Functions

What is the optimum?

- $f_1$ is worse
- $f_2$ is better
Alice & Bob Pay a Fine

- $A: 10\text{₪}, B: 50\text{₪}$ is worse than $A: 0\text{₪}, B: 0\text{₪}$
Alice & Bob Pay a Fine

- $A: 10\$\text{, }B: 50\$ is worse than $A: 0\$, $B: 0$
- What about $A: 0\$, $B: 30$ versus $A: 20\$, $B: 20$?
Alice & Bob Pay a Fine

- $A$: 10₪, $B$: 50₪ is worse than $A$: 0₪, $B$: 0₪.
- What about $A$: 0₪, $B$: 30 versus $A$: 20₪, $B$: 20₪?
- **Leximax**: “Minimize the higher fine.”
Alice & Bob Pay a Fine

- \[ A: 10₪, B: 50₪ \] is worse than \[ A: 0₪, B: 0₪ \]

- What about \[ A: 0₪, B: 30 \] versus \[ A: 20₪, B: 20₪ \]?

- **Leximax**: “Minimize the higher fine.”

- Prefer:
  - \[ A: 20₪, B: 20₪ \] to \[ A: 0₪, B: 30₪ \]
• \$A\$: 10₪, \$B\$: 50₪ is worse than \$A\$: 0₪, \$B\$: 0₪.

• What about \$A\$: 0₪, \$B\$: 30₪ versus \$A\$: 20₪, \$B\$: 20₪?

• **Leximax**: “Minimize the higher fine.”

• Prefer:
  \$A\$: 20₪, \$B\$: 20₪ to \$A\$: 0₪, \$B\$: 30₪.

• **Generalize by**:
  Sort decreasingly and compare lexicographically.
Lexicographic vs Leximax

Lexicographic($f_1, f_2$)

Lexicographic($f_2, f_1$)
Lexicographic vs Leximax

Lexicographic($f_2, f_1$)

Leximax

Lexicographic($f_1, f_2$)
Leximax-optimum

The image shows a bar chart with two sets of bars for each function $f_1$ and $f_2$. The bars are color-coded, with blue for $f_1$ and red for $f_2$. The chart appears to illustrate the comparison between the values of $f_1$ and $f_2$ for different scenarios or conditions.
Leximax-optimum
Leximax-optimum
Leximax-optimum

\[ f_1 \quad f_2 \]

\[ f_1 \quad f_2 \]

\[ f_1 \quad f_2 \]

\[ f_1 \quad f_2 \]

\[ f_1 \quad f_2 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]
SAT-based Leximax Optimisation

$f_1 : x_1 + x_2 + x_3$

$f_2 : x_4 + x_5 + x_6$

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ \max(f_1, f_2) \]
$f_1 : x_1 + x_2 + x_3$

$f_2 : x_4 + x_5 + x_6$
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ o_1 \lor o_4 \Leftrightarrow y_1 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ o_1 \lor o_4 \Leftrightarrow y_1 \]
\[ o_2 \lor o_5 \Leftrightarrow y_2 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \quad \quad o_1 \lor o_4 \Leftrightarrow y_1 \]
\[ f_2 : x_4 + x_5 + x_6 \quad \quad o_2 \lor o_5 \Leftrightarrow y_2 \]
\[ o_3 \lor o_6 \Leftrightarrow y_3 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ \begin{align*}
  o_1 \lor o_4 & \iff y_1 \\
  o_2 \lor o_5 & \iff y_2 \\
  o_3 \lor o_6 & \iff y_3 \\
  y_1 + y_2 + y_3 & = \max(f_1, f_2)
\end{align*} \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ o_1 \lor o_4 \Leftrightarrow y_1 \]
\[ o_2 \lor o_5 \Leftrightarrow y_2 \]
\[ o_3 \lor o_6 \Leftrightarrow y_3 \]

\[ y_1 + y_2 + y_3 = \max(f_1, f_2) \]

**Minimize** \[ y_1 + y_2 + y_3 \]
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ o_1 \lor o_4 \Leftrightarrow y_1 \]
\[ o_2 \lor o_5 \Leftrightarrow y_2 \]
\[ o_3 \lor o_6 \Leftrightarrow y_3 \]

\[ y_1 + y_2 + y_3 = \max(f_1, f_2) \]

**Minimize** \( y_1 + y_2 + y_3 \)

**Optimum** = \( k \)
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ o_1 \lor o_4 \iff y_1 \]
\[ o_2 \lor o_5 \iff y_2 \]
\[ o_3 \lor o_6 \iff y_3 \]

\[ y_1 + y_2 + y_3 = \max(f_1, f_2) \]

Minimize \( y_1 + y_2 + y_3 \)

Optimum = \( k \)

Fix \( y_1 + y_2 + y_3 = k \)
SAT-based Leximax Optimisation

\[ f_1 : x_1 + x_2 + x_3 \]
\[ f_2 : x_4 + x_5 + x_6 \]

\[ o_1 \lor o_4 \iff y_1 \]
\[ o_2 \lor o_5 \iff y_2 \]
\[ o_3 \lor o_6 \iff y_3 \]

\[ y_1 + y_2 + y_3 = \max(f_1, f_2) \]

Minimize \[ y_1 + y_2 + y_3 \]

Optimum = \( k \)

Fix \[ y_1 + y_2 + y_3 = k \]

Repeat with second maximum.
Core-guided Approach

- Major difficulty is the encoding of sorting networks
Core-guided Approach

• Major difficulty is the encoding of sorting networks
  • large CNF encoding
Core-guided Approach

- Major difficulty is the encoding of sorting networks
- **large** CNF encoding
- **difficult** for SAT solvers
• Major difficulty is the encoding of sorting networks
  • large CNF encoding
  • difficult for SAT solvers
• Core-guided:
  • Initially all the variables set to minimum.
  • Permit non-optimality if it appears in a core.
  • Include variables in a core into the sorting network.
  • Augment old sorting network with new variables
  • Or rebuild the network.
Core-guided Approach

- Major difficulty is the encoding of sorting networks
  - **large** CNF encoding
  - **difficult** for SAT solvers
- **Core-guided:**
  - Initially all the variables set to minimum.
Core-guided Approach

- Major difficulty is the encoding of sorting networks
  - large CNF encoding
  - difficult for SAT solvers
- Core-guided:
  - Initially all the variables set to minimum.
  - Permit non-optimality if it appears in a core.
Core-guided Approach

- Major difficulty is the encoding of sorting networks
  - large CNF encoding
  - difficult for SAT solvers
- Core-guided:
  - Initially all the variables set to minimum.
  - Permit non-optimality if it appears in a core.
  - Include variables in a core into the sorting network.
Core-guided Approach

- Major difficulty is the encoding of sorting networks
  - **large** CNF encoding
  - **difficult** for SAT solvers
- **Core-guided:**
  - Initially all the variables set to minimum.
  - Permit non-optimality if it appears in a core.
  - Include variables in a core into the sorting network.
    - **Augment** old sorting network with new variables
Core-guided Approach

- Major difficulty is the encoding of sorting networks
  - **large** CNF encoding
  - **difficult** for SAT solvers
- Core-guided:
  - Initially all the variables set to minimum.
  - Permit non-optimality if it appears in a core.
  - Include variables in a core into the sorting network.
    - Augment old sorting network with new variables
    - Or Rebuild the network.
Evaluation

- Instances:
  Package Upgradeability benchmarks from the Mancoosi project
Evaluation

• Instances:
  Package Upgradeability benchmarks from the Mancoosi project

• mccc (ILP-based algorithm)
Evaluation

- Instances:
  Package Upgradeability benchmarks from the Mancoosi project
- **mccs** (ILP-based algorithm) with CPLEX, Gurobi, SCIP, Cbc, GLPK, lpsolve
Evaluation

• Instances:
  Package Upgradeability benchmarks from the Mancoosi project
• `mccs` (ILP-based algorithm) with
  CPLEX, Gurobi, SCIP, Cbc, GLPK, lpsolve
Evaluation

- Instances:
  Package Upgradeability benchmarks from the Mancoosi project
- mccs (ILP-based algorithm) with
  CPLEX, Gurobi, SCIP, Cbc, GLPK, lpsolve
Evaluation

- Instances:
  Package Upgradeability benchmarks from the Mancoosi project
- \texttt{mccs} (ILP-based algorithm) with CPLEX, Gurobi, SCIP, Cbc, GLPK, lpsolve
- \texttt{packup} (SAT-based algorithms) with CaDiCaL
Evaluation

- **Instances:**
  Package Upgradeability benchmarks from the Mancoosi project
- **mccs (ILP-based algorithm) with**
  CPLEX, Gurobi, SCIP, Cbc, GLPK, lpsolve
- **packup (SAT-based algorithms) with CaDiCaL**
  - various non-core-guided search (binary, linear, etc.)
  - various core-guided search differing on how to augment the sorting network
Comparison — SAT-based

Packup - SAT-based algorithms

Instances vs Time in seconds

- core-merge-dc
- core-merge
- core-rebuild-dc
- core-rebuild-incr-dc
- binary
- linear-us
- core-static
- linear-su
- core-rebuild
- core-rebuild-incr
Evaluation — Package Upgradeability

Package Upgradeability

Instances
0 1000 2000 3000
0
20
40
60
80
100
120
140
160
180
Time in seconds
0
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800

Gurobi
CPLEX
core-merge-dc
Cbc
SCIP
GLPK
Ip_solve
Evaluation — SAT Competition

MaxSAT Evaluation

Time in seconds

Instances

core-merge-dc
Gurobi
CPLEX
Conclusions and Future Work

- Solving Multi-Objective Optimization for Leximax
Conclusions and Future Work

- Solving Multi-Objective Optimization for **Leximax**
- **SAT**-based solving
Conclusions and Future Work

- Solving Multi-Objective Optimization for **Leximax**
- **SAT**-based solving
- **Core**-based solving
Conclusions and Future Work

• Solving Multi-Objective Optimization for Leximax
• SAT-based solving
• Core-based solving
• Evaluation on Package Upgradeability
Conclusions and Future Work

- Solving Multi-Objective Optimization for Leximax
- **SAT**-based solving
- **Core**-based solving
- Evaluation on Package Upgradeability
- Core-guided the best out of the SAT-based