**SOLVING WCSP BY EXTRACTION OF MINIMAL UNSATISFIABLE CORES**

CHRISTOPHE LEcoutre, NICOLAS PARIS, OLIVIER ROUSSEL AND SÉBASTIEN TABARY

{lecoutre.paris.roussel.tabary}@cril.univ-artois.fr

---

**Introduction**

- **Context:**
  - CSP and Weighted CSP frameworks
  - WCSP algorithms are often more complex than their CSP counterparts (due to management of costs)

- **Goal:**
  - Benefit from efficient CSP algorithms developed for more than two decades

- **Principle:**
  - Solve WCSP by iteratively generating and solving classical CSPs (greedy approach)
  - The sequence of CSPs is enumerated according to an increasing cost order related to the WCSP
  - Minimal Unsatisfiable Cores (MUC) identify the soft constraints whose costs must be increased

---

**Algorithm**

```plaintext
VMERI(W, J)
forall i ∈ constraints(W) do
  if sol ∩ i = ∅ then
    return sol
  else
    P ← IsCSP_i(W, f); sol ← solveCSP(P);
    if sol ∩ i = ∅ then
      return sol
    else
      M ← extractMUC(P);
      W' ← restrict(W, M);
      f ← relax(W', f);
    endif
  endif
endforall
```

- **IsCSP_i(W, f):**
  - Translates WCSP into CSP, converting soft constraints into hard constraints according to a front f
  - Considering a soft constraint v and the front f, a hard constraint is obtained by selecting as allowed tuples in v the tuples of the layers whose index is less than or equal to f(v)
  - Representation: extension, positive table (supports) / negative table (conflicts)
  - Default cost Layer forbidden (resp. allowed) → hard positive (resp. negative) constraint

- **extractMUC(P):**
  - A dichotomic approach is used to extract MUCs of unsatisfiable CSPs

- **relax(W', f):**
  - MUCs are broken by generating successors of the front considering only the constraints of MUCs

---

**Background**

**CSP framework:**
- A CSP is satisfiable iff it admits at least one solution
- An Unsatisfiable Core is an unsatisfiable subset of constraints
- A core is a Minimal Unsatisfiable Core (MUC) iff each strict subset is satisfiable

**WCSP framework:**
- Extension of CSP
- \((X, C, k): C\) is a set of soft constraints (cost functions), \(k > 0\) is either a natural integer or +∞
- \(\forall u, v \in [0, \ldots, K], u \oplus v = \min(k, a + b)\)
- **Goal:**
  - find a complete instantiation with minimal optimization problem
  - The current methods to solve WCSPs: branch and bound tree search combined with the use of soft local consistencies (EDAC, etc. by cost transfer) for estimating minimal costs of sub-problems during search

---

**Layers and fronts**

- Focus on soft table constraints (explicit and implicit tuples), but the method can be easily extended to other kinds of constraints
- A layer contains all tuples having the same cost
- A front (represented by an array \(f\)) maps each constraint of a WCSP to one of its layers

- **Cost of a front:** sum of costs associated with the selected layers

---

**Experiments**

<table>
<thead>
<tr>
<th>Instances</th>
<th>GMR</th>
<th>AbsCon</th>
<th>EDAC</th>
<th>ToolBar2</th>
<th>INCOOP</th>
<th>EDAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>spot5/spot5-401</td>
<td>CPU</td>
<td>4.99</td>
<td>1</td>
<td>1</td>
<td>217</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>118</td>
<td>114</td>
<td>114</td>
<td>114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spot5/spot5-412</td>
<td>CPU</td>
<td>15.8</td>
<td>1</td>
<td>1</td>
<td>316</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>51,401</td>
<td>41,399</td>
<td>32988</td>
<td>47,389</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spot5/spot5-501</td>
<td>CPU</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>8226</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>22,266</td>
<td>20,258</td>
<td>21266</td>
<td>25,268</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spot5/spot5-509</td>
<td>CPU</td>
<td>32.2</td>
<td>1</td>
<td>1</td>
<td>37462</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>57,669</td>
<td>49,475</td>
<td>57462</td>
<td>46,477</td>
<td></td>
<td></td>
</tr>
<tr>
<td>spot5/spot5-1001</td>
<td>CPU</td>
<td>142.5</td>
<td>1</td>
<td>1</td>
<td>88277</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>481,206</td>
<td>757,265</td>
<td>95277</td>
<td>757,265</td>
<td></td>
<td></td>
</tr>
<tr>
<td>color/graph-05</td>
<td>CPU</td>
<td>16.6</td>
<td>1</td>
<td>1</td>
<td>0.06</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>221</td>
<td>245</td>
<td>241</td>
<td>221</td>
<td></td>
<td></td>
</tr>
<tr>
<td>color/acnn-06-20</td>
<td>CPU</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>67.9</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>1,403</td>
<td>1,594</td>
<td>1,466</td>
<td>1,603</td>
<td></td>
<td></td>
</tr>
<tr>
<td>color/acnn-07</td>
<td>CPU</td>
<td>209.9</td>
<td>1</td>
<td>1</td>
<td>209.9</td>
<td></td>
</tr>
<tr>
<td>UB</td>
<td>426,423</td>
<td>312,295</td>
<td>39006</td>
<td>505,731</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CPU time (in seconds) to prove optimality on various selected instances (time-out of 600 seconds set per instance, \(\uparrow\): time-out reached)

---

**Conclusion and future work**

- **Conclusion**
  - Original greedy approach: solve WCSP through successive resolutions of CSPs
  - Focus the cost increase on the sole constraints in the Minimal Unsatisfiable Cores extracted
  - Promising results when compared to other state-of-the-art approaches

- **Future work**
  - Complete approach based on the same principles (work in progress)