A SAT-Based Approach to Cooperative Path-Finding Using All-Different Constraints

Charles University in Prague
Czech Republic

Pavel Surynek

Kobe University
Japan

SoCS 2012
Cooperative Path-finding (CPF)

- plan movements of agents in **space** and **time**
  - **time** – discrete ⇒ time steps
  - **space** – abstract ⇒ graph $G=(V,E)$
- requirements
  - all agents reach a given **goal vertex**
  - agents do **not collide** with each other
    (move only to vacant vertices)

Initial state

```
A  v1  v4  v7  v8  v9  v9
B  v2  v5  v8
C  v3  v6  v9
```

Goal state

```
A  v1  v4  v7  v8  v9  v9
B  v2  v5  v8
C  v3  v6  v9
```

Set of **agents** = {1,2,3}

- plan for **agent A** = [v1, v4, v7, v8, v9, v9, v9]
- plan for **agent B** = [v2, v2, v1, v4, v7, v8, v8]

Time step: 1 2 3 4 5 6 7

makespan = 7
Current Techniques / Our Approach

- fast, complete
  - long makespan

- polynomial time
  - sub-optimal

  + SAT Solver

  + encoding of CPF

  + optimization strategy

- relatively fast
  - incomplete

- optimal makespan
  - slow

search based
- sub-optimal

search based
- optimal

= Our new approach – iCBOBOP

- (quickly) find sub-optimal solution
- replace sub-sequences with makespan-optimal sub-solutions
- repeat the process

Pavel Surynek, 2012
SAT Encoding of CPF

- encoding for the **fixed makespan** \( m \)
- encode state at each time-step
  - multi-value state variables \( \Rightarrow \) bit-vectors

\[
\begin{align*}
\mathcal{L}_1^A &= v_1 \\
\mathcal{L}_1^B &= v_2 \\
\mathcal{L}_1^C &= v_3 \\
&\ldots\\
\mathcal{L}_m^A &= v_9 \\
\mathcal{L}_m^B &= v_8 \\
\mathcal{L}_m^C &= v_7
\end{align*}
\]

- state at time-step 1 = initial state
- state at time-step \( i \)
- state at time-step \( i+1 \)
- state at time-step \( m \) = goal state

agent \( a \) can move into an unoccupied vertex only:
\( \mathcal{L}_i^a \neq \mathcal{L}_j^b \) for all \( b \neq a \)

agents move along edges of the graph

at most one agent is located in each vertex: All-Different(\( \mathcal{L}_i^a \mid a \in \text{Agents} \)) for \( i=1,\ldots,m \)
Optimization Strategy - iCOBOPT

- for a fixed makespan $m$ find the longest sub-sequence of the original solution that can be replaced with corresponding optimal sub-solution of makespan $m$

- iCOBOPT is similar to turbo-fullstep
Experimental Results and Comparison

- setup: $G=(V,E) = 4$-connected grid

random initial and goal arrangement of agents

Grid 16×16 | few agents
---
<table>
<thead>
<tr>
<th>Number of agents</th>
<th>Base solution</th>
<th>WHCA*</th>
<th>Inverse</th>
<th>All-different</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>N/A</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Base solutions generated by the BIBOX algorithms (Surynek, 2009)

Grid 16×16 | many agents
---
<table>
<thead>
<tr>
<th>Number of agents</th>
<th>Base solution</th>
<th>Inverse</th>
<th>All-different</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6/6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>21/21</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>15/15</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>18/18</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Number of time steps

<table>
<thead>
<tr>
<th>Number of time steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 4 8 12 16 20 24 28 32 36 40</td>
</tr>
</tbody>
</table>

Optimal makespan

<table>
<thead>
<tr>
<th>Number of agents</th>
<th>4-connected grid 16×16</th>
<th>4-connected grid 16×16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Optimal makespan</td>
<td>SATPLAN Runtime (s)</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0.68</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>195.5</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>1396.07</td>
</tr>
<tr>
<td>16</td>
<td>N/A</td>
<td>Out of memory</td>
</tr>
</tbody>
</table>

Computed makespan

<table>
<thead>
<tr>
<th>Number of agents</th>
<th>4-connected grid 16×16</th>
<th>4-connected grid 16×16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Computed makespan</td>
<td>INVERSE Runtime (s)</td>
</tr>
<tr>
<td>1</td>
<td>6/6</td>
<td>0.074</td>
</tr>
<tr>
<td>4</td>
<td>21/21</td>
<td>319.785</td>
</tr>
<tr>
<td>8</td>
<td>15/15</td>
<td>152.625</td>
</tr>
<tr>
<td>16</td>
<td>18/18</td>
<td>1833.080</td>
</tr>
</tbody>
</table>

Pavel Surynek, 2012
Conclusions and Related Works

- Good performance on graphs with dense population agents
- Sometimes optimal solution can be found
- Encoding, sub-optimal algorithm, and optimization strategy can be improved independently

References

