Near Optimal Cooperative Path Planning in Hard Setups through Satisfiability Solving

Pavel Surynek

Charles University in Prague, Czech Republic
and Kobe University, Japan
Problem of Cooperative Path-planning (CPP)

- **Abstraction** for tasks of motion of multiple (autonomous or passive) entities in a certain environment (real or virtual).
  - Entities are given an **initial** and a **goal** arrangement in the environment.
  - We need to **plan movements of entities in time**, so that entities reach the goal arrangement while **physical limitations are observed**.

- **Physical limitations** are:
  - Entities must **not collide with each other**.
  - Entities must **not collide with obstacles** in the environment.

- Cooperative path-planning is also known as:
  - pebble motion on a graph
  - path-planning for multiple robots

- Certain slight variations in the definition allows higher parallelism.
A popular moving puzzle, that can be abstracted as the problem of cooperative path-planning is known as **Lloyd’s fifteen.**

- Entities are represented by *pebbles/agents* labeled by numbers.

The environment is modeled as an **undirected graph** where *vertices represent locations* in the environment occupied by agents and *edges* enable agents to go to the *neighboring location.*

**Formal definition** of the task of CPP

It is a quadruple \( \Pi = (G, A, S_A^0, S_A^+) \), where:

- \( G=(V,E) \) is an **undirected graph,**
- \( A = \{a_1,a_2,...,a_\mu\} \), where \( \mu<|V| \) is a set of agents,
- \( S_A^0: A \rightarrow V \) is a uniquely invertible function determining the **initial arrangement of agents** in vertices of \( G \), and
- \( S_A^+: A \rightarrow V \) is a uniquely invertible function determining the **goal arrangement of agents** in vertices of \( G \).
Time is discrete in the model. **Time steps** and their ordering is isomorphic to the structure of natural numbers.

The **dynamicity** of the task is as follows:

- An agent occupying a vertex at time step \( i \) can move into a neighboring vertex (the move is finished at time step \( i+1 \)) if the target vertex is **unoccupied** at time step \( i \) and **no other agent** is moving simultaneously into the same target vertex.

For the given \( \Pi = (G, A, S_A^0, S_A^+) \), we need to find:

- A sequence of moves for every agent such that dynamicity constraint is satisfied and every agent reaches its goal vertex.

---

**Solution** of an instance of cooperative path-planning on a graph with \( A=\{1,2,3\} \):

- \( M_1=[v_1, v_4, v_7, v_8, v_9, v_9, v_9] \)
- \( M_2=[v_2, v_2, v_1, v_4, v_7, v_8, v_8] \)
- \( M_3=[v_3, v_3, v_3, v_2, v_1, v_4, v_7] \)

**Time step:** 1 2 3 4 5 6 7

**makespan** = 7
Motivation

- Container rearrangement (entity = \textit{container})
- Heavy traffic (entity = \textit{automobile} (in jam))
- Data transfer (entity = \textit{data packet})
- Generalized lifts (entity = \textit{lift})
Is the task of CPP easy or hard?

- **Basic** variant of the task is easy to solve (makespan sub-optimal solution):
  - There exists an algorithm with worst case time complexity of $O(|V|^3)$ that generates solutions of the makespan $O(|V|^3)$ for any instance of CPP on $G=(V,E)$ (Kornhauser et al., 1984).

- If we want a solution that has the makespan as short as possible the complexity increases:
  - The optimization variant of the CPP problem is NP-hard (Ratner a Warmuth, 1986)
  - Shown for the generalized Lloyd’s puzzle (known as $(N^2-1)$-puzzle).

- We focused on generating and improving sub-optimal solutions towards optimal makespan.
COBOPT – CPP as Propositional Satisfiability

- Suppose that we are able to construct a propositional formula such that
  - It is satisfiable iff there exists a solution to CPP of a given makespan

- Suppose that we are provided with makespan suboptimal solution (base solution – can be generated in polynomial time)
  - we can find makespan optimal replacement of the given sub-sequence of the base solution using:
    - propositional satisfiability solving + binary search (or some other type of search where query = SAT solving for the given makespan)
Inverse Encoding of CPP

- **Makespan** $m$ is given
  - encode states of the planning world at time steps 1, 2, ..., $m$
  - state at the time **step 1** is enforced to be equal to the **initial state**
  - state at the time **step m** is enforced to be equal to the **goal state**
- Inverse encoding encodes “**what agent** is located in the given vertex”
- The state at the given time step $i$ is described by the following **integer variables** for each $v \in V$:
  - $A_v^i \in \{0, 1, 2, ..., \mu\}$ with the interpretation that
    - $A_v^i = j$ iff the agent $a_j$ is located in $v$ at the time step $i$
    - $A_v^i = 0$ iff there is no agent in $v$
  - $T_v^i \in \{0, 1, 2, ..., 2\deg(v)\}$ with the interpretation that
    - $0 < T_v^i \leq \deg(v)$ iff the agent goes out of $v$ into $(T_v^i)$-th neighbor
    - $\deg(v) \leq T_v^i \leq 2\deg(v)$ iff the agent goes into $v$ from $(\deg(v)-T_v^i)$-th neighbor
    - $T_v^i = 0$ iff no-operation is selected for $v$
- **Plus constraints to enforce valid transitions between states**
Translating to Propositional Satisfiability

- Each integer variable is encoded as a bit-vector where each bit is represented by a propositional variable
  - for example $A_v^i \in \{0, 1, 2, \ldots, \mu\}$ is encoded using $\lceil \log_2(\mu+1) \rceil$ propositional variables
  - extra states induced by the upper integer part are forbidden

- **Notice**: a bit-vector must take some of the values from its domain
  - each $T_v^i$ must be assigned a value ... in each vertex it must be decided what action is selected (no-op, incoming, outgoing)
  - ensures that agents do not collide with each other and maintains the frame

- Implications of the form $T_v^i = \text{constant} \implies A_u^{i+1} = \text{constant}$
  - translated using auxiliary propositional variables
All-Different Encoding of CPP

- If the environment contains few agents relatively to its size
  - inverse encoding contains lot of variables for empty space
- All-Different encoding encodes “where is the given agent”
- The state at the given time step $i$ is described by the following integer variables for each $a \in A$:
  - $L_a^i \in \{1, 2, \ldots, |V|\}$ with the interpretation that $L_a^i = j$ iff the agent $a$ is located in the $j$-th vertex of the graph $G$
- The requirement that there is at most one agent per vertex is modeled as All-Different($L_{a_1}^i, L_{a_2}^i, \ldots, L_{a_\mu}^i$)
- Other constraints are more complicated
  - it is necessary to express that agents can move along edges only
  - and that target vertex of the movement must be empty
- Augmenting by heuristics
  - some vertices are unreachable by the agent in the given time step
## Encoding Size Comparison

- Two setups grid of size 8x8 and 16x16
- random initial and goal arrangement of agents

<table>
<thead>
<tr>
<th></th>
<th>in the 4-connected grid 8x8</th>
<th>Number of layers</th>
<th>SATPLAN encoding</th>
<th>SASE encoding</th>
<th>INVERSE encoding</th>
<th>ALL-DIFFERENT encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Variables</td>
<td>Clauses</td>
<td>Variables</td>
<td>Clauses</td>
<td>Variables</td>
<td>Clauses</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>5864</td>
<td>55330</td>
<td>11386</td>
<td>53143</td>
<td>5400</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>10022</td>
<td>165660</td>
<td>19097</td>
<td>105724</td>
<td>5920</td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>14471</td>
<td>356410</td>
<td>26857</td>
<td>168875</td>
<td>5920</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>30157</td>
<td>1169198</td>
<td>51662</td>
<td>372140</td>
<td>8122</td>
</tr>
<tr>
<td>24</td>
<td>10</td>
<td>43451</td>
<td>2473813</td>
<td>73101</td>
<td>588886</td>
<td>8122</td>
</tr>
<tr>
<td>32</td>
<td>14</td>
<td>99398</td>
<td>8530312</td>
<td>157083</td>
<td>1385010</td>
<td>12396</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>in the 4-connected grid 16x16</th>
<th>Number of layers</th>
<th>SATPLAN encoding</th>
<th>SASE encoding</th>
<th>INVERSE encoding</th>
<th>ALL-DIFFERENT encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Variables</td>
<td>Clauses</td>
<td>Variables</td>
<td>Clauses</td>
<td>Variables</td>
<td>Clauses</td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>69704</td>
<td>746562</td>
<td>137406</td>
<td>677737</td>
<td>60755</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>65365</td>
<td>995507</td>
<td>134482</td>
<td>712352</td>
<td>46904</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>30157</td>
<td>1169198</td>
<td>342100</td>
<td>2347456</td>
<td>61154</td>
</tr>
<tr>
<td>32</td>
<td>4*</td>
<td>Out of memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>4*</td>
<td>288498</td>
<td>2716096</td>
<td>13672</td>
<td>143104</td>
<td>197888</td>
</tr>
<tr>
<td>64</td>
<td>4*</td>
<td>357762</td>
<td>3783672</td>
<td>13672</td>
<td>134912</td>
<td>265280</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Variables</th>
<th>Clauses</th>
<th>Variables</th>
<th>Clauses</th>
<th>Variables</th>
<th>Clauses</th>
<th>Variables</th>
<th>Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>6</td>
<td>561210</td>
<td>5913320</td>
<td>14700</td>
<td>189440</td>
<td>510464</td>
<td>2446912</td>
<td></td>
</tr>
</tbody>
</table>
Compared against WHCA*

WHCA* is decoupled
- often produces near makespan optimal
Makespan Comparison – grid 16x16

Makespan | Grid 16x16 | few agents

Makespan | Grid 16x16 | many agents
Parallelism Increasing

### Grid 8x8
- **Original Parallellism**
- **Optimized Parallellism**

### Grid 16x16
- **Original Parallellism**
- **Optimized Parallellism**

- Number of moves vs. Parallelism vs. |Agents|
Concluding Remarks

- Improving sub-optimal solutions of cooperative path-planning by modeling the problem as propositional satisfiability.

- COBOPT: short subsequences of a sub-optimal solution are replaced by the makespan optimal ones.

- Two encodings (and its variants)
  - Inverse encoding
    - better in densely populated environments
  - All-Different encoding
    - better in sparsely populated environments

- COBOPT solution optimization together with both encodings represents state-of-the-art in generating short solutions to CPP