Reactive Task and Motion Planning under Temporal Logic Specifications

Shen Li*, Daehyung Park*, Yoonchang Sung*, Julie A. Shah, Nicholas Roy
Massachusetts Institute of Technology

* These authors contributed equally to this work.
Robot-assisted packing

https://www.roadaffair.com/best-travel-organizers/
Task-and-motion planning

Region 1
Region 2
Region 3
Human can relocate objects

Region 1

Region 2

Region 3

Sense + replan
Human can add objects

Region 1

Region 2

Sense + replan

Region 3
Human can remove objects

Region 1

Region 2

Region 3

Sense + replan
Human can add regions

Region 1

Region 2

Region 4

Region 3

Sense + replan
Problem definition

• Task and motion planning
• Environmental changes
  • Unexpected changes made by humans in human-robot interaction
Environmental changes

- Object relocation
- Object addition and removal
- Region addition
Environmental changes

• Object relocation
• Object addition and removal
• Region addition

Computationally expensive to sense and replan on the fly.
Place all objects in the region on the right.
Replanning on the fly is computationally expensive.
Hierarchical architecture for efficient replanning and execution to handle environmental changes.
Hierarchical System Design
To generate reactive behavior
Eventually always keep all objects in region 2.

\[ \mathcal{FG}(\text{all\_obj\_in\_r}_2) \]
LTL specification $\rightarrow$ Buchi automaton (BA) (known)

$$FG(\text{all\_obj\_in\_r2})$$

```
qb_init
```

```
<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
</table>
```

```
all_obj_in_r2
```

```
qb_accept
```

```
all_obj_in_r2
```

```latex
<latexit TFKi+cAQTCQzt0JyhSUm2sSYhvD5KfyfXO7ZTslG5/uF6tEkjizYAJtgGzigDKrgDNRAHRBwC+7BI3iy7qwH69l6GbdmrMnMOvgB6/UDVaWebg==</latexit>
```
Robot, workspace → Transition system (TS)

- $s_1 = o_1r_1, o_2r_1, o_3r_1$
- Move $o_1$ from $r_1$ to $r_2$
- Motion cost

- $s_2 = o_1r_2, o_2r_1, o_3r_1$
- Move $o_2$ from $r_1$ to $r_2$
- Motion cost

- $s_3 = o_1r_2, o_2r_2, o_3r_1$
- Move $o_3$ from $r_1$ to $r_2$
- Motion cost

- $s_4 = o_1r_2, o_2r_2, o_3r_2$

...
Robot, workspace $\rightarrow$ Transition system (TS)

LTL specification $\rightarrow$ Buchi automaton (BA)
Graph construction

Robot, workspace → Transition system (TS)
                  x
LTL specification → Buchi automaton (BA)
                  =
Product automaton (PA)
Graph construction

Robot, workspace

LTL specification

Transition system (TS)

Buchi automaton (BA)

Product automaton (PA)

A* graph search algorithm

Perception system

Current state

High-level task plan
Graph construction

Robot, workspace → Transition system (TS) → Buchi automaton (BA) → Product automaton (PA)

LTL specification

Perception system → Current state → A* graph search algorithm

High-level task plan
• Move o1 from r1 to r2
• Move o2 from r1 to r2
• Move o3 from r1 to r2
Graph construction

- Transition system (TS)
- Buchi automaton (BA)
- Product automaton (PA)

LTL specification

A* graph search algorithm

Perception system

Current state

Dynamically reconfigurable Behavior tree

Subtree 1  Subtree 2  ......  Subtree n

A high-level task plan
A high-level task plan:

- Move o1 from r1 to r2
- Move o2 from r1 to r2
- Move o3 from r1 to r2
Dynamically reconfigurable Behavior tree

Robot, workspace → Transition system (TS) → Buchi automaton (BA) → Product automaton (PA)

LTL specification → A* graph search algorithm

Perception system → Current state

Dynamically reconfigurable Behavior tree

Subtree 1 → Subtree 2 → …… → Subtree n

Current state

Low-level primitive controller

A high-level task plan
Dynamically reconfigurable Behavior tree

Graph construction
- Transition system (TS)
- Buchi automaton (BA)
- Product automaton (PA)

LTL specification

A* graph search algorithm

Perception system
- Current state

Robot, workspace
- Current state

Low-level primitive controller

A high-level task plan
Efficiently handle environmental changes

1. Hierarchical system design
2. Algorithmic design
Hierarchical system design to efficiently handle environmental changes

**Graph construction**

- Robot, workspace
- LTL specification

**Transition system (TS)**

**Buchi automaton (BA)**

**Product automaton (PA)**

**A* graph search**

- Perception system
- Current state

**Dynamically reconfigurable Behavior tree**

- Subtree 1
- Subtree 2
- .......
- Subtree n

**Low-level primitive controller**

**Current state**

**A high-level task plan**
Hierarchical system design to efficiently handle environmental changes

Graph construction

Transition system (TS) x Buchi automaton (BA) = Product automaton (PA)

A* graph search

Perception system

Current state

Dynamically reconfigurable Behavior tree

Subtree 1, Subtree 2, ..., Subtree n

Low-level primitive controller
Handle **different environmental changes** using **different components** in the hierarchical architecture

Hierarchical System Design for Efficient Interference Handling

- Robot, workspace
- Transition system (TS)
- Buchi automaton (BA)
- Product automaton (PA)
- Perception system
- Current state
- A* graph search
- A high-level task plan
- Dynamically reconfigurable Behavior tree
- Subtree 1
- Subtree 2
- ...... Subtree n
- Current state
- Low-level primitive controller
Environmental changes

- Object relocation
- Object addition and removal
- Region addition
Dynamically reconfigurable Behavior tree

Subtree 1  Subtree 2  ......  Subtree n

Current state

Low-level primitive controller

• Object relocation 1
**Behavior tree**

Subtree 1: Move o1 from r1 to r2

Subtree 2: Move o2 from r1 to r2

Subtree 3: Move o3 from r1 to r2
Behavior tree

Subtree 1: Move o1 from r1 to r2
Subtree 2: Move o2 from r1 to r2
Subtree 3: Move o3 from r1 to r2
Behavior tree

Subtree 1: Move o1 from r1 to r2
Subtree 2: Move o2 from r1 to r2
Subtree 3: Move o3 from r1 to r2
Behavior tree

Subtree 1:
Move o1 from r1 to r2

Subtree 2:
Move o2 from r1 to r2

Subtree 3:
Move o3 from r1 to r2
Behavior tree

Subtree 1:
Move o1 from r1 to r2

Subtree 2:
Move o2 from r1 to r2

Subtree 3:
Move o3 from r1 to r2
Dynamically reconfigurable Behavior tree

Subtree 1  Subtree 2  .......  Subtree n

Current state

Low-level primitive controller

• Object relocation 1
Dynamically reconfigurable Behavior tree

- Subtree 1
- Subtree 2
- .......
- Subtree n

Current state

Low-level primitive controller

Reuse planning experience
- Subtree

Object relocation 1
A* graph search

Perception system

Current state

A high-level task plan

Dynamically reconfigurable Behavior tree

Subtree 1  Subtree 2  ......  Subtree n

Current state

Low-level primitive controller

- Object relocation 2
Behavior tree

Subtree 1: Move o1 from r1 to r2
Subtree 2: Move o2 from r1 to r2
Subtree 3: Move o3 from r1 to r2
Behavior tree

Subtree 1: Move o1 from r1 to r2
Subtree 2: Move o2 from r1 to r2
Subtree 3: Move o3 from r1 to r2
Behavior tree

Subtree 1: Move o1 from r1 to r2
Subtree 2: Move o2 from r1 to r2
Subtree 3: Move o3 from r1 to r2
A* graph search

Perception system

Current state

A high-level task plan

Dynamically reconfigurable Behavior tree

• Object relocation 2

Subtree 1 Subtree 2 ...... Subtree n

Current state

Low-level primitive controller
A* graph search

Perception system

Current state

High-level task plan

Dynamically reconfigurable Behavior tree

Subtree 1  Subtree 2  ......  Subtree n

Current state

Low-level primitive controller

Reuse planning experience
- Motion cost

A high-level task plan

Object relocation 2

Motion cost
A* graph search

Perception system

Current state

A high-level task plan

Reuse planning experience

• Motion cost

Reuse planning experience

• Subtree

Dynamically reconfigurable Behavior tree

Subtree 1  Subtree 2  ......  Subtree n

Current state

Low-level primitive controller

Object relocation 2
Graph construction

- Robot, workspace
- Transition system (TS)
- Buchi automaton (BA)
- Product automaton (PA)
- LTL specification

A* graph search

Perception system

Current state

Dynamically reconfigurable Behavior tree

- Subtree 1
- Subtree 2
- ……
- Subtree n

A high-level task plan

Low-level primitive controller

- Object addition / removal
- Region addition
Graph construction

Transition system (TS)

Buchi automaton (BA)

Product automaton (PA)

Robot, workspace

LTL specification

Perception system

Current state

A* graph search

Dynamically reconfigurable Behavior tree

Subtree 1

Subtree 2

......

Subtree n

Low-level primitive controller

• Object addition / removal

• Region addition

• Object relocation 1

• Object relocation 2

A high-level task plan

Current state
Experiments

\[ FG(all\_obj\_in\_r_2) \]

- Random changes to env
  - Repositioning 1 object
  - Removing 1 object
  - Adding 1 object
  - Adding 1 region
Experiments

$FG(all_{obj\_in\_r2})$
## Types of changes to the environment

<table>
<thead>
<tr>
<th></th>
<th>Success</th>
<th>Init plan time (s)</th>
<th>Total replan time (s)</th>
<th># replan</th>
<th>Complete time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object relocation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>68.39 ± 1.00</td>
<td><strong>0.69 ± 0.43</strong></td>
<td>0.80, 0</td>
<td><strong>466.50 ± 34.80</strong></td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>67.19 ± 0.04</td>
<td>33.50 ± 15.26</td>
<td>0.63, 0</td>
<td>530.17 ± 19.04</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.36 ± 0.04</td>
<td>30.40 ± 13.06</td>
<td>0.47, 0</td>
<td>561.57 ± 27.29</td>
</tr>
<tr>
<td><strong>Object removal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>67.62 ± 0.22</td>
<td><strong>0.10 ± 0.02</strong></td>
<td>1.93, 1</td>
<td><strong>394.48 ± 13.14</strong></td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>67.34 ± 0.07</td>
<td>52.57 ± 8.18</td>
<td>1.30, 1</td>
<td>536.25 ± 31.49</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.57 ± 0.08</td>
<td>60.53 ± 9.16</td>
<td>1.40, 1</td>
<td>478.43 ± 17.54</td>
</tr>
<tr>
<td><strong>Object addition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>67.78 ± 0.32</td>
<td><strong>20.61 ± 0.86</strong></td>
<td>1.37, 1</td>
<td><strong>647.00 ± 24.67</strong></td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>69.16 ± 1.29</td>
<td>89.60 ± 8.73</td>
<td>1.13, 1</td>
<td>734.74 ± 33.32</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.60 ± 0.10</td>
<td>116.16 ± 16.35</td>
<td>1.33, 1</td>
<td>720.81 ± 25.03</td>
</tr>
</tbody>
</table>
## Proposed algorithm: A* + reuse experience

<table>
<thead>
<tr>
<th>Method</th>
<th>3-block + tray</th>
<th>Success</th>
<th>Init plan time (s)</th>
<th>Total replan time (s)</th>
<th># replan</th>
<th>Comple time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object relocation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>68.39 ± 1.00</td>
<td><strong>0.69 ± 0.43</strong></td>
<td>0.80, 0</td>
<td><strong>466.50 ± 34.80</strong></td>
<td></td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>67.19 ± 0.04</td>
<td>33.50 ± 15.26</td>
<td>0.63, 0</td>
<td>530.17 ± 19.04</td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.36 ± 0.04</td>
<td>30.40 ± 13.06</td>
<td>0.47, 0</td>
<td>561.57 ± 27.29</td>
<td></td>
</tr>
<tr>
<td><strong>Object removal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>67.62 ± 0.22</td>
<td><strong>0.10 ± 0.02</strong></td>
<td>1.93, 1</td>
<td><strong>394.48 ± 13.14</strong></td>
<td></td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>67.34 ± 0.07</td>
<td>52.57 ± 8.18</td>
<td>1.30, 1</td>
<td>536.25 ± 31.49</td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.57 ± 0.08</td>
<td>60.53 ± 9.16</td>
<td>1.40, 1</td>
<td>478.43 ± 17.54</td>
<td></td>
</tr>
<tr>
<td><strong>Object addition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>67.78 ± 0.32</td>
<td><strong>20.61 ± 0.86</strong></td>
<td>1.37, 1</td>
<td><strong>647.00 ± 24.67</strong></td>
<td></td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>69.16 ± 1.29</td>
<td>89.60 ± 8.73</td>
<td>1.13, 1</td>
<td>734.74 ± 33.32</td>
<td></td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.60 ± 0.10</td>
<td>116.16 ± 16.35</td>
<td>1.33, 1</td>
<td>720.81 ± 25.03</td>
<td></td>
</tr>
</tbody>
</table>
Hierarchy ~ increased efficiency

<table>
<thead>
<tr>
<th>3-block + tray</th>
<th>Success</th>
<th>Init plan time (s)</th>
<th>Total replan time (s)</th>
<th># A* plan</th>
<th>Complete time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object relocation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>68.39 ± 1.00</td>
<td>0.69 ± 0.43</td>
<td>0.80, 0</td>
<td>466.50 ± 34.80</td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>67.19 ± 0.04</td>
<td>33.50 ± 15.26</td>
<td>0.63, 0</td>
<td>530.17 ± 19.04</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.36 ± 0.04</td>
<td>30.40 ± 13.06</td>
<td>0.47, 0</td>
<td>561.57 ± 27.29</td>
</tr>
<tr>
<td><strong>Object removal</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>67.62 ± 0.22</td>
<td>0.10 ± 0.02</td>
<td>1.93, 1</td>
<td>394.48 ± 13.14</td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>67.34 ± 0.07</td>
<td>52.57 ± 8.18</td>
<td>1.30, 1</td>
<td>536.25 ± 31.49</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.57 ± 0.08</td>
<td>60.53 ± 9.16</td>
<td>1.40, 1</td>
<td>478.43 ± 17.54</td>
</tr>
<tr>
<td><strong>Object addition</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A*, exp</td>
<td>30/30</td>
<td>67.78 ± 0.32</td>
<td>20.61 ± 0.86</td>
<td>1.37, 1</td>
<td>647.00 ± 24.67</td>
</tr>
<tr>
<td>A*</td>
<td>30/30</td>
<td>69.16 ± 1.29</td>
<td>89.60 ± 8.73</td>
<td>1.13, 1</td>
<td>734.74 ± 33.32</td>
</tr>
<tr>
<td>Dijkstra</td>
<td>30/30</td>
<td>67.60 ± 0.10</td>
<td>116.16 ± 16.35</td>
<td>1.33, 1</td>
<td>720.81 ± 25.03</td>
</tr>
</tbody>
</table>

Number of calls to A* graph search
10x (1x when human is in the scene)
Reactive Task and Motion Planning under Temporal Logic Specifications

- Task and motion planning under environmental changes
  - Hierarchical system design for reactive behavior generation
- **Efficiently** handle environmental changes
  1. Hierarchical system design
  2. Algorithmic design
Reactive Task and Motion Planning under Temporal Logic Specifications

Shen Li*, Daehyung Park*, Yoonchang Sung*, Julie A. Shah, Nicholas Roy
Massachusetts Institute of Technology

* These authors contributed equally to this work.