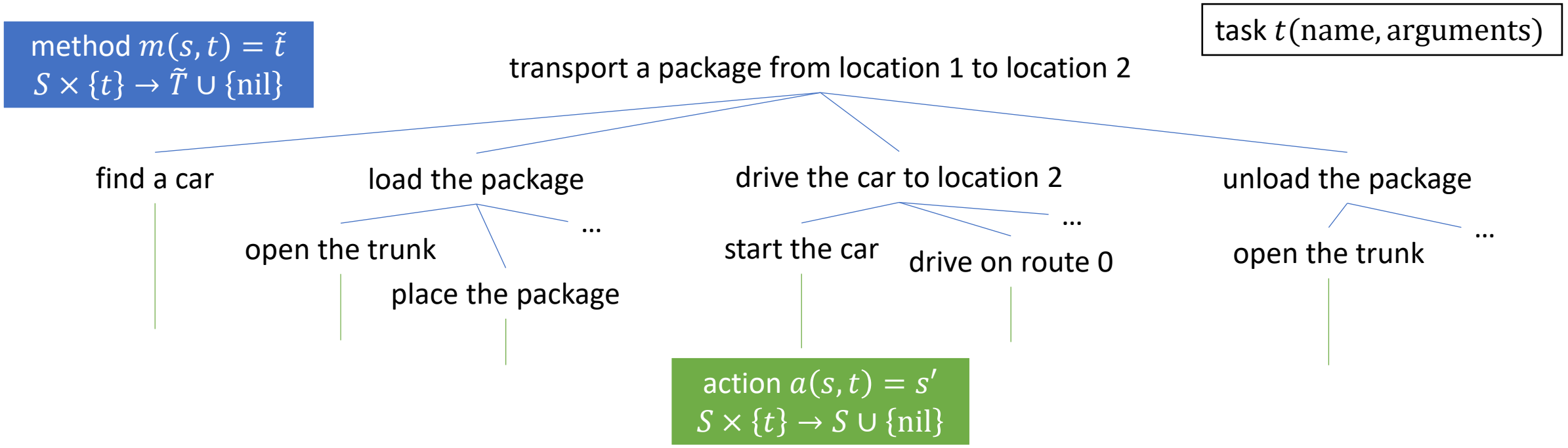
Task Modifiers for HTN Planning and Acting

Weihang Yuan
Hector Munoz-Avila
Lifang He
Lehigh University

Venkatsampath Gogineni
Sravya Kondrakunta
Michael Cox
Wright State University

Hierarchical Task Networks (HTNs)

- HTN planning decomposes compound tasks into primitive tasks, which define actions that change the world state.

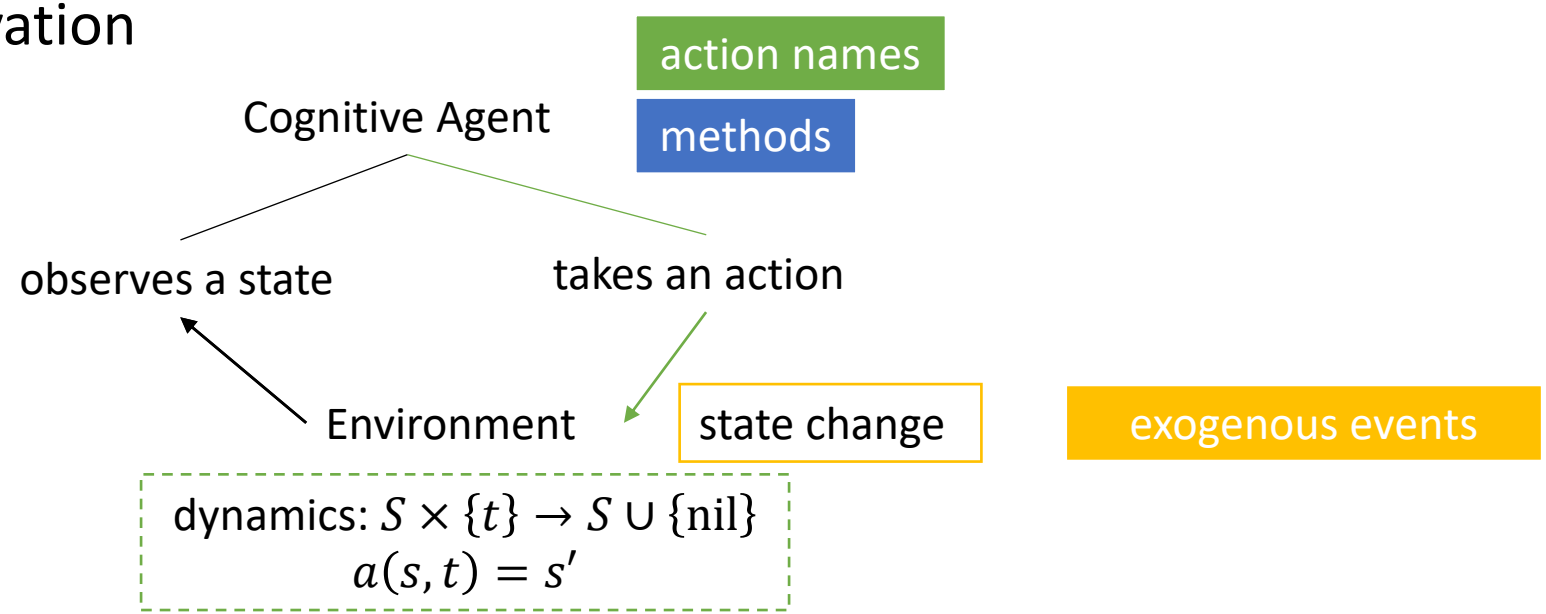


HTN Planning

- An HTN planning problem $(s, \tilde{t}, D)$
  - s is a state
  - $\tilde{t} = (t_1, \dots, t_n)$ is a task list
  - D consists of a set of actions and a set of methods
- A plan $\pi = (a_1, \dots, a_m)$ is a solution if
  - If $\tilde{t} = \emptyset$, then $\pi = \emptyset$
  - If $\tilde{t} = \emptyset$,
    - If t_1 is primitive and a_1 is applicable and $(a_2, \dots, a_m)$ is a solution for $(s', (t_2, \dots, t_n), D)$
    - If t_1 is compound and there exists an applicable method m and π is a solution for $(s, (\text{subtasks}, t_2, \dots, t_n), D)$

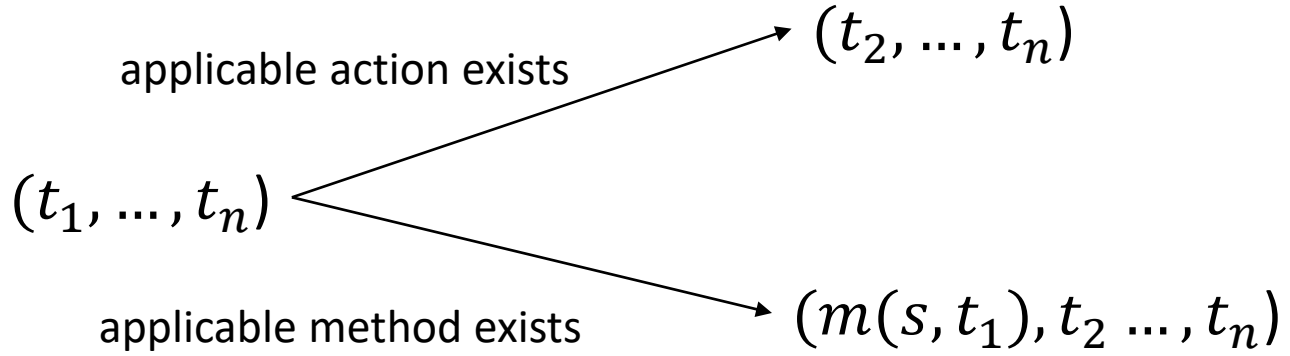
Motivation

- In some problems, environmental dynamics are not fully observable.
- $(s, \tilde{t}, D')$
  - D = a set of action names and a set of methods
  - s is an observation



Task Modifiers

- Originally, an agent's task list can only be modified in two ways:



- Task modifier: $S \times \tilde{T} \rightarrow \tilde{T}$

$$TM(s, \tilde{t}) = \tilde{t}'$$

  - TMs provide an additional way to modify the task list.
  - TMs receive a task list as input whereas methods receive a single task.

Task Modifiers

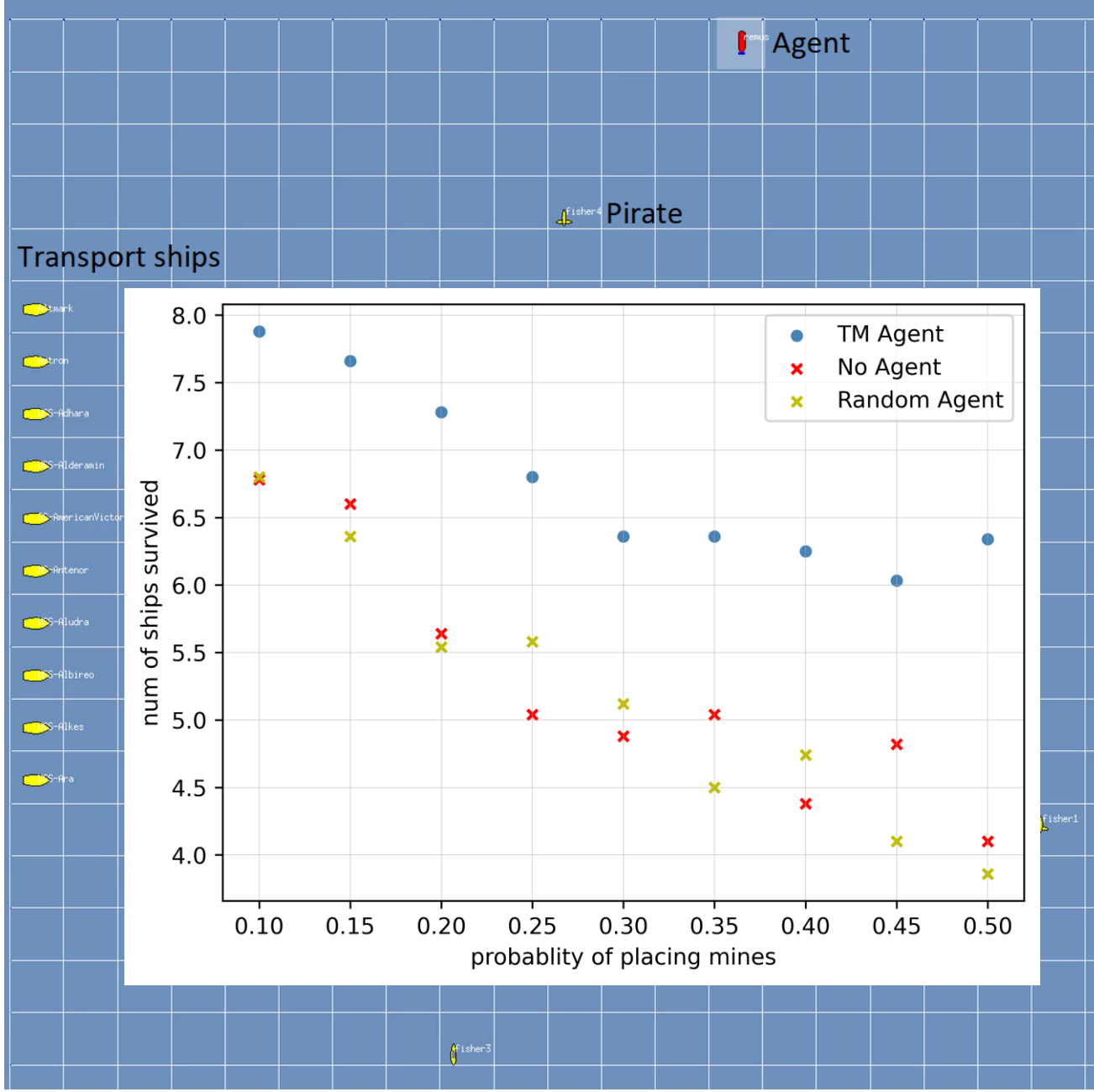
- An algorithm that integrates SHOP with a task modifier and interleaves planning and execution

```
1:  procedure PLAN-ACT-TM(t̃, D)
2:      observe s
3:      return SEEK-PLAN-ACT-TM(s, t̃, D)

4:  procedure SEEK-PLAN-ACT-TM(s, t̃, D)
5:      if t̃ = ∅ or the episode terminates then
6:          return s
7:      t ← the first task in t̃; R ← the remaining tasks
8:      if t is primitive then
9:          if there is an action a(s, t) ≠ nil then
10:             apply a
11:             observe s′
12:             R ← TM(s′, R)
13:             return SEEK-PLAN-ACT-TM(s′, R, D)
14:         else
15:             return nil
16:     else
17:         for every method m(s, t) ≠ nil do
18:             s ← SEEK-PLAN-ACT-TM(s, (m(s, t), R), D)
19:             if s ≠ nil then
20:                 return s
21:         return nil
```

Experiments

- Minefield: maximize the number of transport ships that survive.
- The agent has no direct knowledge of
  - Identity of the pirate
  - Locations of mines
- The agent has a predefined TM
- The baseline has a random TM



Summary

- Describe an extension to HTN called task modifiers as a solution to a type of domains
- Describe an algorithm that integrates task modifiers and SHOP
- Empirically demonstrate the feasibility of this approach

Thank you