Review of Hierarchical Multi-Agent Reinforcement Learning

Team JRL
Jesse Meyer, Ryan Praeuner, Luke Vanderbeek

Paper used as source

Ghavamzadeh, M., S. Mahadevan, and R. Makar (2006). Hierarchical Multi-Agent Reinforcement Learning, *Journal of Autonomous Agents and Multiagent Systems*, **13**: 197-229.

Outline

- Introduction
- HRL Framework
- Pros and Cons
- Cooperative HRL algorithm
- Cooperative HRL algorithm with communication
- Conclusion

Introduction

- HRL accelerates learning
  - Cooperative subtasks
  - Highest level of hierarchy
  - Primitive action complexity
- Trash collection problem
  - Cooperative HRL
  - Selfish multi-agent HRL
  - Single-agent HRL
  - Q-learning

Challenges

- Curse of dimensionality
  - Parameters to be learned vs number of agents
- Partial observability
  - Actions of other agents
  - Communication
- HRL
  - Task hierarchies to scale reinforcement learning
  - Task structure restricts space

Algorithms

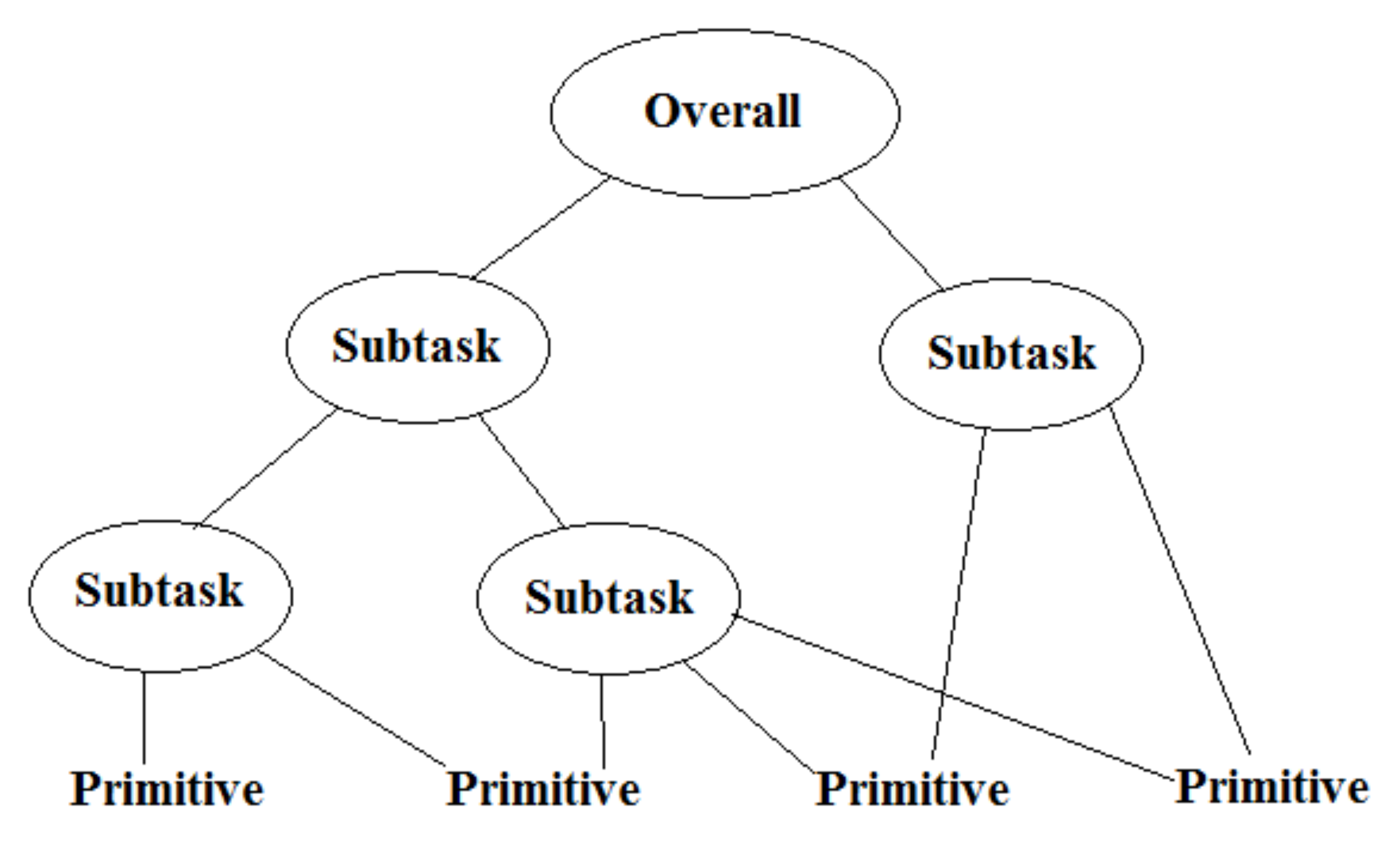
- Cooperative HRL
    - Homogeneous
    - Decentralized learning
        - Perform subtasks
        - Order of execution
        - Coordination with other agents

- COM-Cooperative HRL
    - Adds communication level below cooperation level
    - Optimize action and communication

HRL Framework for multi-agents

- Lets agents use hierarchical structure to learn tasks
- Task split into different levels:
  - Primitive Tasks
  - Subtasks
  - Overall task
- Subtask sharing
  - Only one agent has to learn each
- Can use graph to represent task relations
  - state abstraction

Task Chart



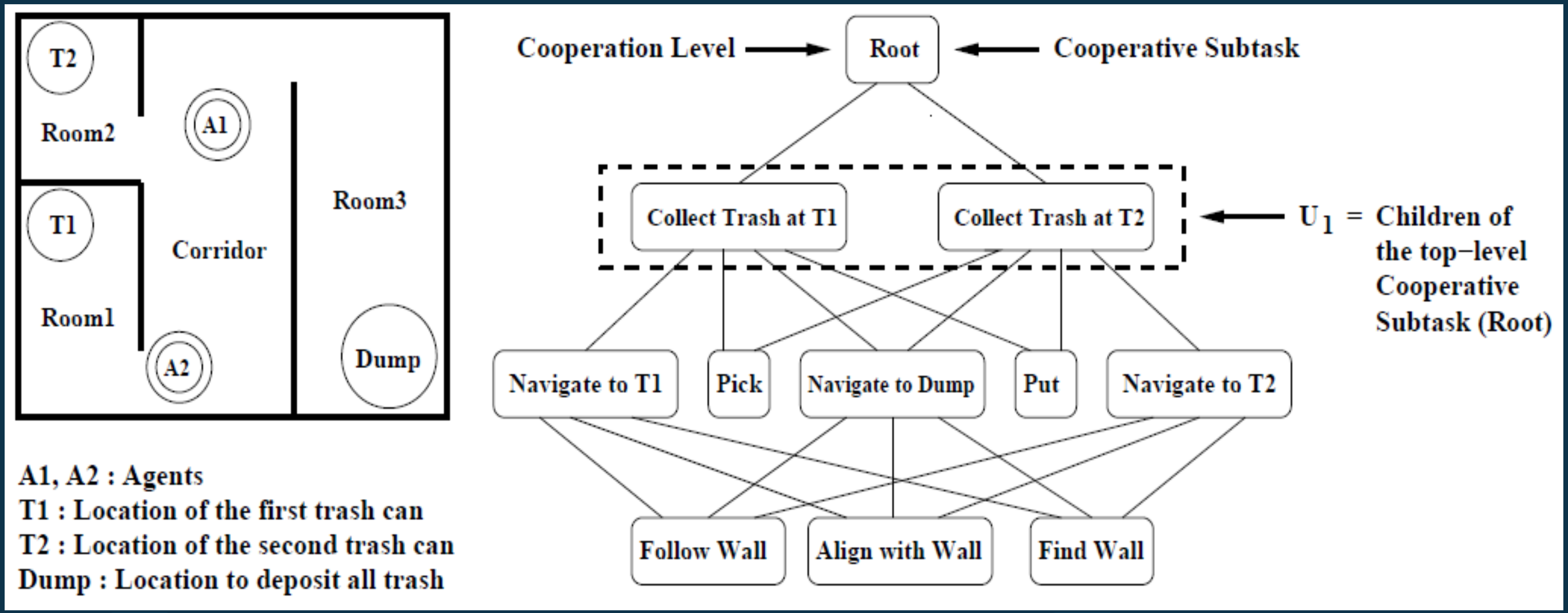
Overall
Subtask
Subtask
Subtask
Subtask
Primitive
Primitive
Primitive
Primitive

Example - Trash Pickup Robots

- Simple task that could use HRL
- As described in the paper
  - To pick up trash and take it to dump zone
  - Can be parallelized by more then one agent(A1, A2)
  - More then one pick up spot(T1, T2)
  - One dump zone(Dump)
  - For example in an office

- Subtasks
  - Navigate to T1, T2 or Dump
  - When to perform Pick or Put action
  - Order of other subtasks

Example Diagram



Semi-Markov Decision Processes

- Decisions only made at discrete points in time
- State of the system may change between decisions
- Decision epochs

- Used for multi-agent system domains
  - Assume agents cooperative
  - agent's actions effect others decisions
  - actions may terminate at different times

- Termination strategies
  - synchronous - T_{any} or T_{all}
  - asynchronous - $T_{continue}$

Multi-Agent Setup

- Agents are homogeneous
  - share same task hierarchy
  - heterogeneous more complicated
- System designer makes task chart
  - Could automate this
- Cooperative subtasks are set before hand
- High level of coordination
  - agents look less are lower details

Pros and Cons of Co-op Multi-agent

- Pros
  - scales large state spaces down
  - fast cooperation
    - only done at high level(s)
  - Less communication needed
- Cons
  - Low cooperative level can cause none optimal solution
  - Storing only local state information is sub-optimal

Cooperative HRL Algorithm

- In this algorithm:
  - an agent starts from the root task and chooses a subtask until it reaches a primitive action.
  - It executes primitive action in the current state
  - Receives reward
  - Observes resulting state
  - Updates the value function of primitive subtask
- assumes zero communication cost

Experimental Results

- The size of the state space would grow to: 124 locations * 124 locations * 4 objects * 4 objects = 240,000 states with multiple agents

- 124 locations * 3 objects * 3 objects = 1116 states with a single agent

- Agents learn a specific policy.

- Number of steps greatly reduced.

Learned Policy for Agent 1

root
navigate to T1
go to location of T1 in room 1
pick trash from T1
navigate to Dump
exit room 1
enter room 3
go to location of Dump in room 3
put trash collected from T1 in Dump
end

Learned Policy for Agent 2

root
navigate to T2
go to location of T2 in room 2
pick trash from T2
navigate to Dump
exit room 2
enter room 3
go to location of Dump in room 3
put trash collected from T2 in Dump
end

Required Steps

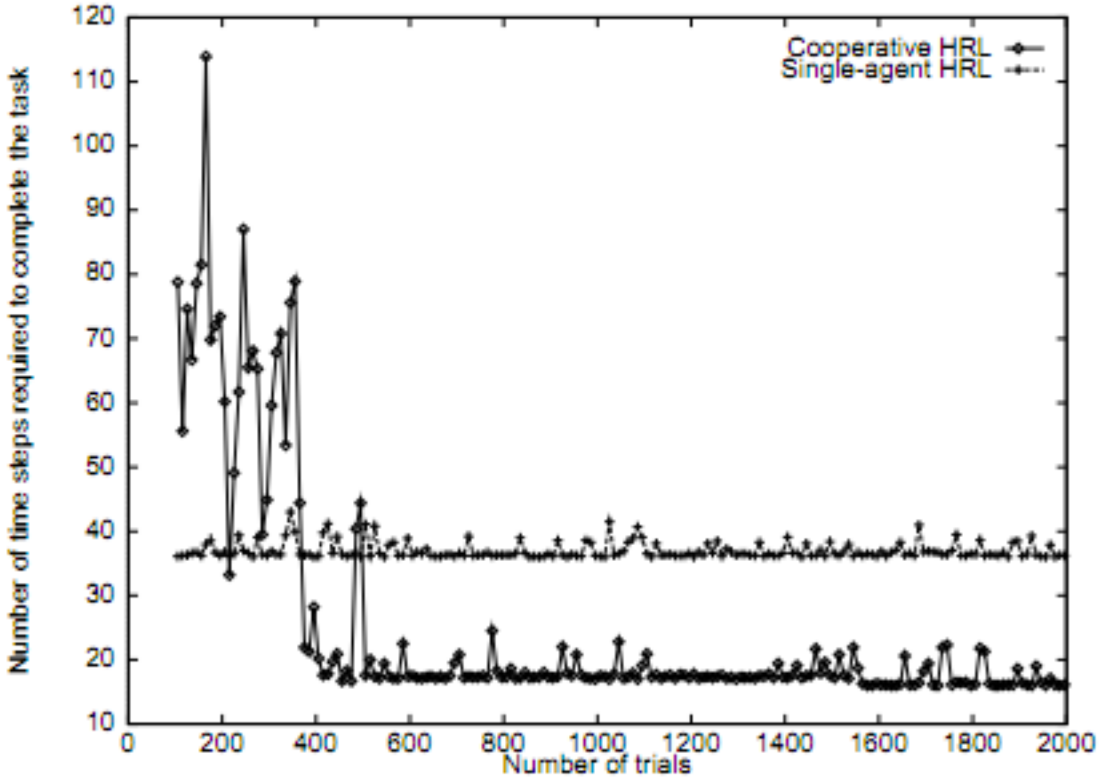


Figure 4. This figure shows that the *Cooperative HRL* algorithm learns the trash collection task with fewer number of steps than the single-agent HRL algorithm.

Cooperative HRL with communication

- Same steps in algorithm with extra communication level
- In the real world, communication is not free.
- Communication usually consists of three steps: send, answer, and receive.
  - send: agent j decides if communication is necessary, performs a communication action, and sends a message to agent i
  - answer: agent i receives the message from agent j, updates its local information using the content of the message, and sends back the answer.
  - receive: agent j receives the answer, updates local information, and decides on action.

Cooperative HRL with communication

- Generally there are two types of messages in a communication framework: request and inform.
  - Tell: agent j sends and inform message to agent i
  - Ask: agent j sends request message to agent i, i responds with inform message
  - Sync: agent j sends inform message to agent i, which is answered with an inform message

Cooperative HRL with communication

- Agents must learn to use communication optimally.
  - compare expected values
- If no communication, acts like selfish agent.
- Communication:
  - sends request message to all agents
  - respond with actions in an inform message

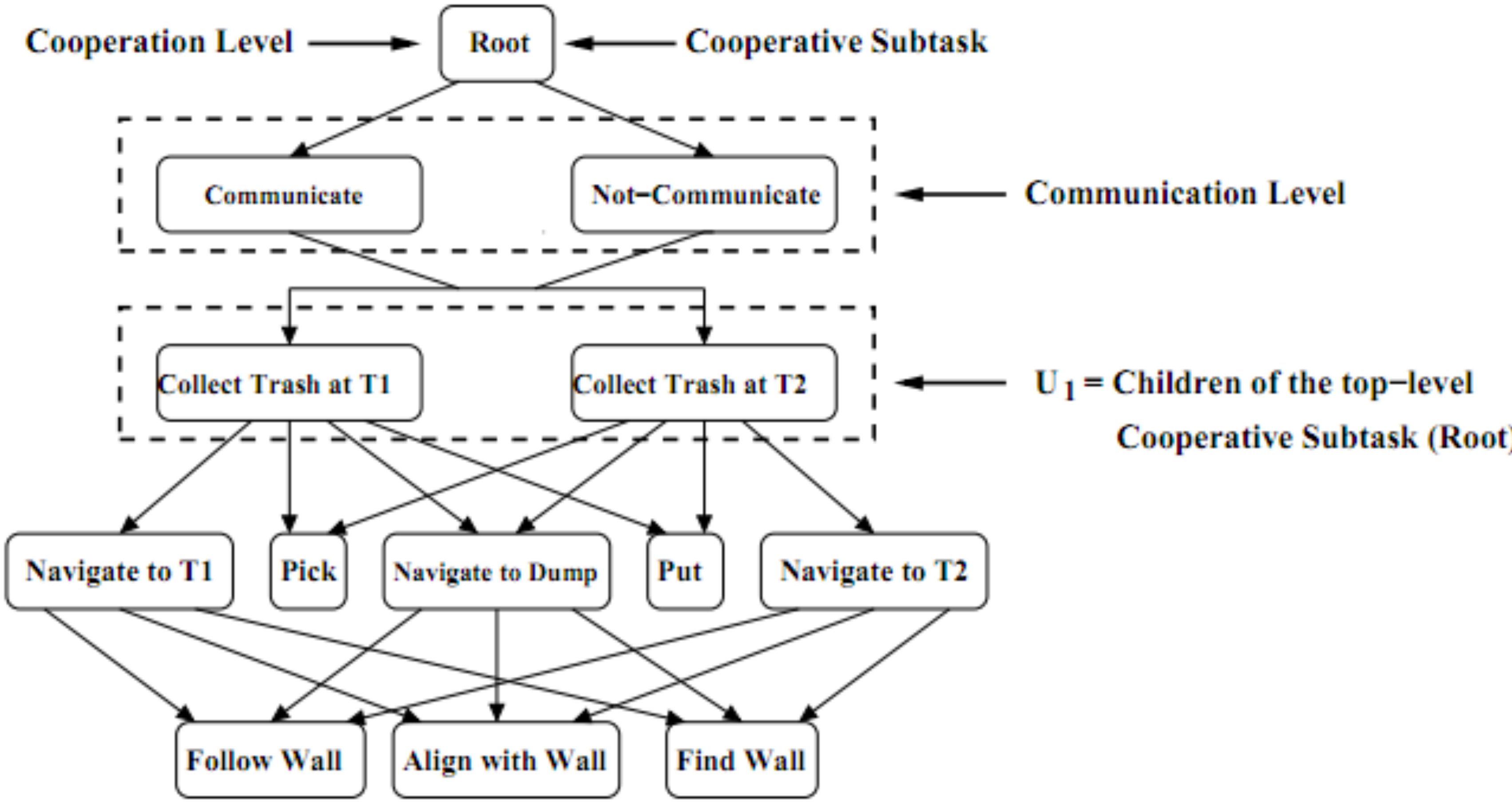
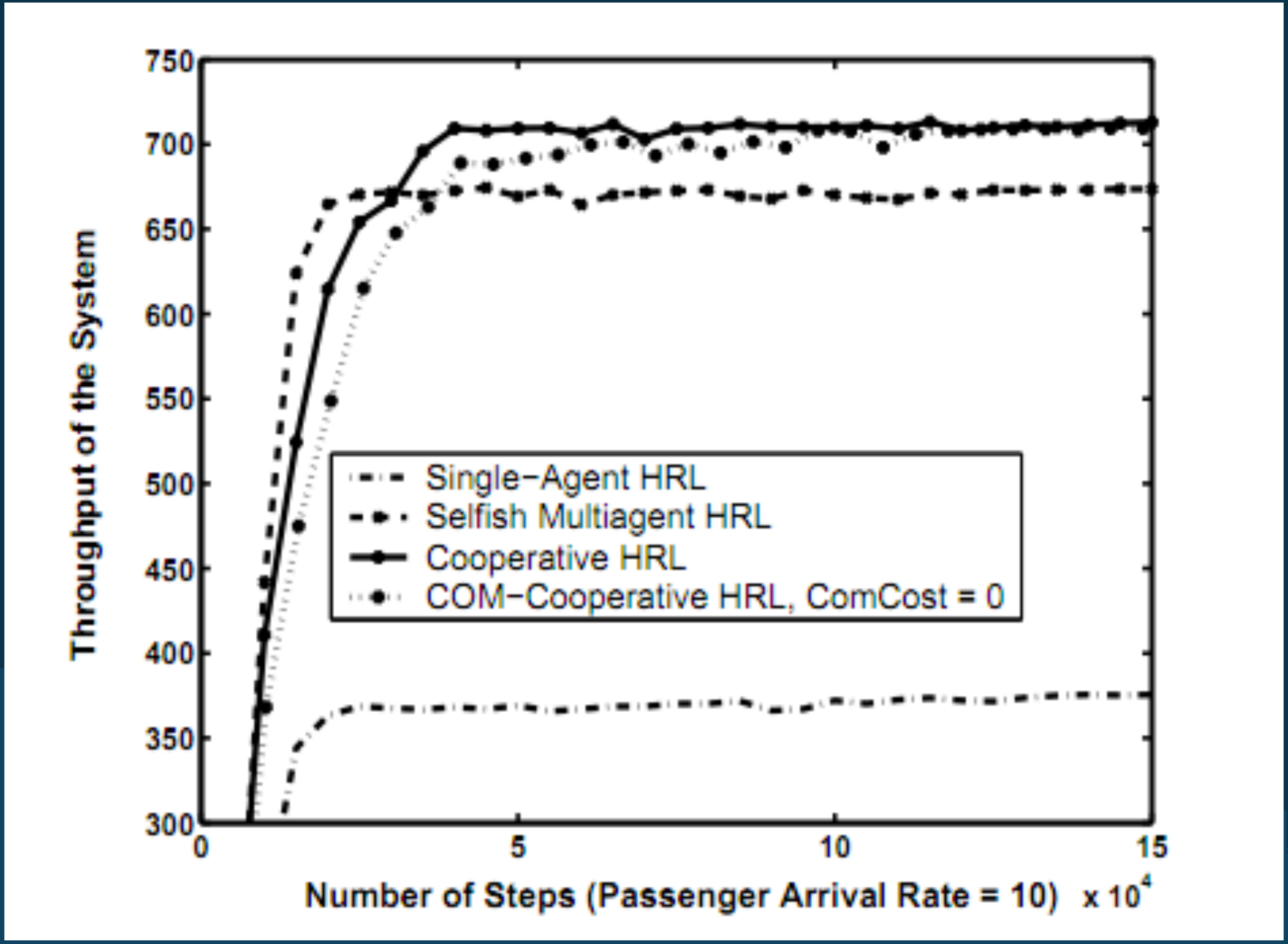


Figure 12. Task graph of the trash collection problem with communication actions.

Experimental Results

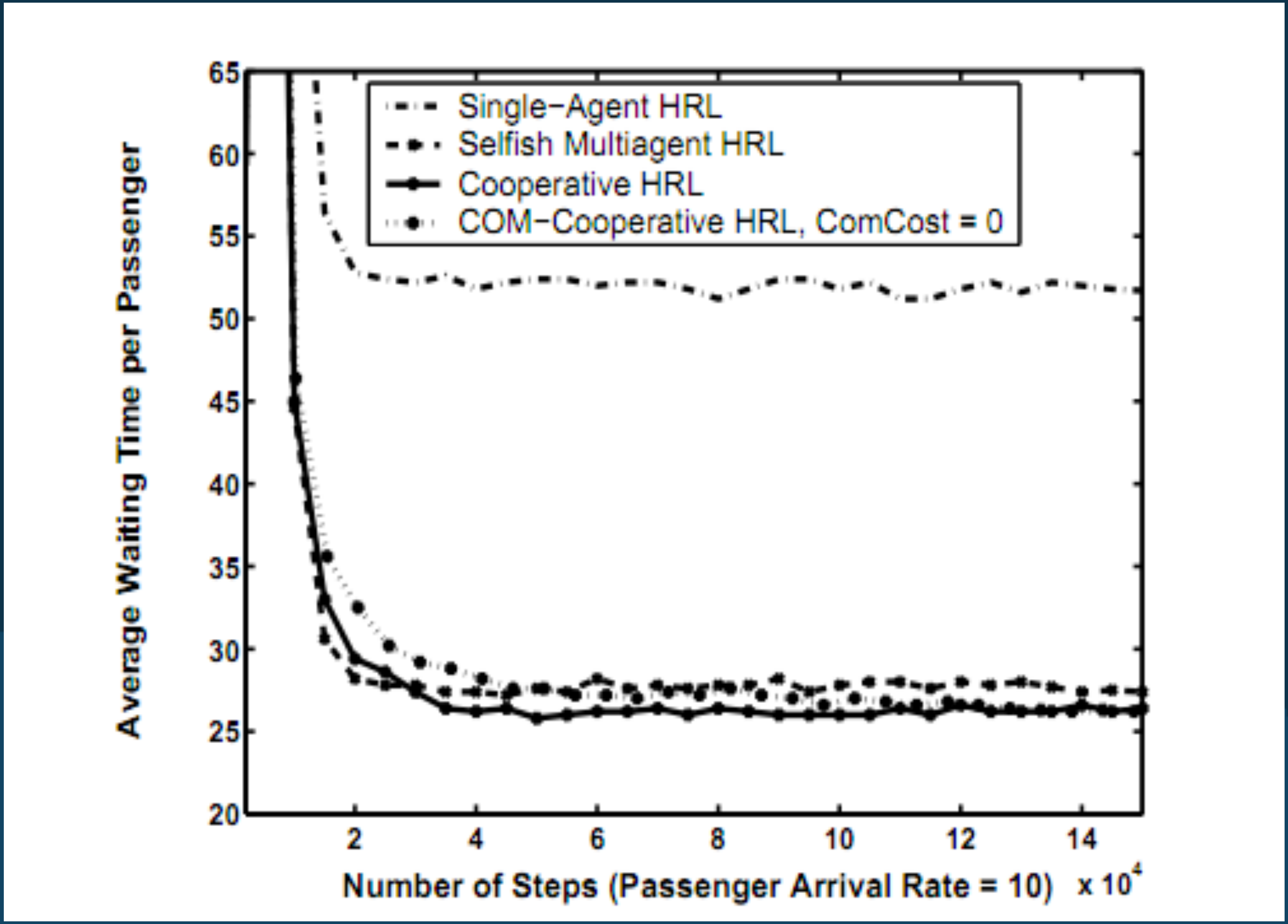
- Taxi example:
  - Two taxis
  - passengers arrive at stations
- On average, has a higher throughput and lower waiting time.

Throughput



Throughput of the System
750
700
650
600
550
500
450
400
350
300
0
5
10
15
Number of Steps (Passenger Arrival Rate = 10) x 10^4
Single−Agent HRL
Selfish Multiagent HRL
Cooperative HRL
COM−Cooperative HRL, ComCost = 0

Waiting Time



Conclusion

- If you want more accuracy, use the communication model.
- Graph to represent sub tasks has to be made, this can be a huge downside
- The key idea is that coordination skills are learned much more efficiently if agents have a hierarchical representation of the task structure.

Questions?