Multi-agent Cooperation for Warehouse Delivery

Shuo Jiang

January 2020

Contents

1 Introduction

2 Method

Our methods employed three level hierarchies. At the highest level, it is a task allocation, which assign task to each agent. The middle level, after each agent receive the task, they will perform a sequence of macro actions to achieve the goal given by the task, which can be independent when the task can be solved alone or coordinated when the task requires coordination. According to if the task requires coordination, the macro actions are conducted synchronously or asynchronously. The transition of macro actions are represented by an FSA. At low level, the macro action will be translated into a sequence of primitive actions by planning algorithms.

2.1 Task Allocation

2.2 Macro Action

The valid set of macro action is predefined. In our scenario, we define such macro actions idle, go to box, lift(independent), lift(joint), deliver(independent), de-

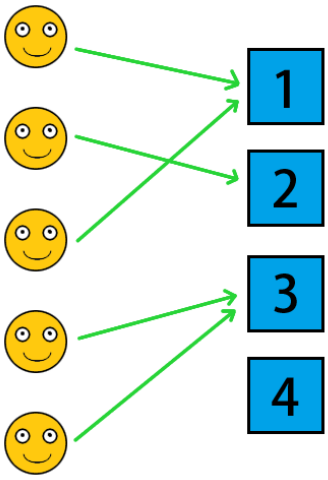

Figure 1: Task allocation

liver(joint). The policy over such macro action set is represented by an finite state automata (FSA). The macro actions are nodes in the FSA and the transition is shown in Figure (1). The difference between independent and joint is when in independent mode, the agent runs asynchronously and independent, but in joint mode, the agent has to communicate and coordinate with the agents assigned to the same task.

An example is shown in Figure (2), where the left two agents are independently assigned to task 1 and 2, so they behaves in transition of independent loop as in bottom boxes. The right two agents are assigned to the same task, so their transition loop is joint which include the red nodes, and when they are in the red nodes, they have to be coordinated with valid communication channel.

The time sequence is shown in Figure (3). We can see all the yellow nodes in independent mode are running asynchronously and the red nodes which are joint nodes are running synchronously.

2.3 Primitive Action

The primitive actions are given by planners such as A-star algorithm for navigation task.

2.3.1 Action Buffer

We set up a buffer for each agent. As the result of planning can be a long sequence of actions, executing such sequence normally will not reach the goal state in such dynamic environment. However redo planning at each time step

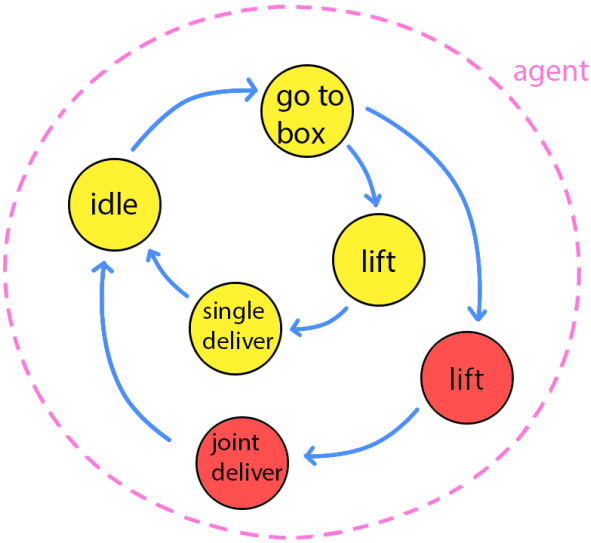

Figure 2: FSA transition

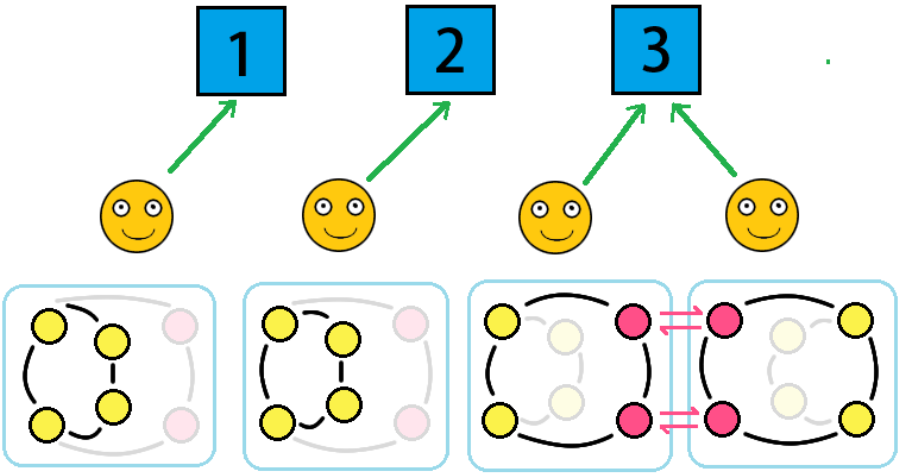

Figure 3: FSA transition (global)

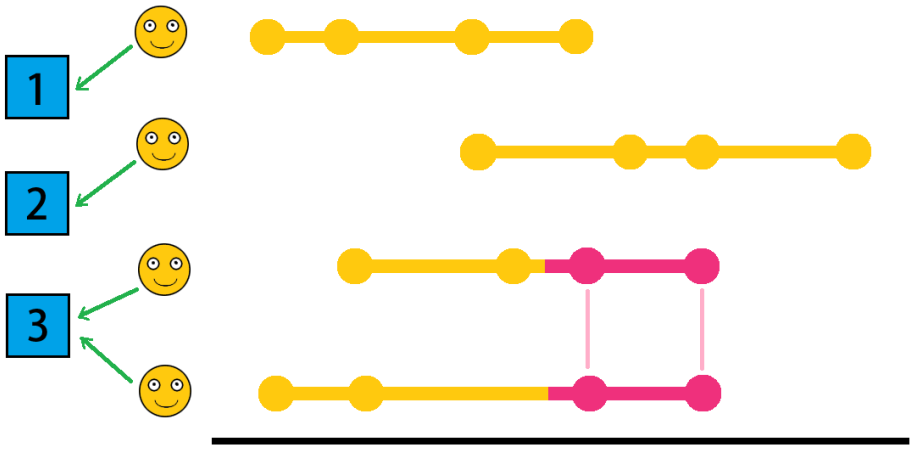

Figure 4: FSA transition time Series for single allocation

can also be redundant. So we only keep the nearest n steps of action planned in the buffer and re-plan when all the actions have been executed. Figure (4) shows such scheme that each step one action will be executed and pop out off the buffer, and the buffer is re-filled when it is empty.

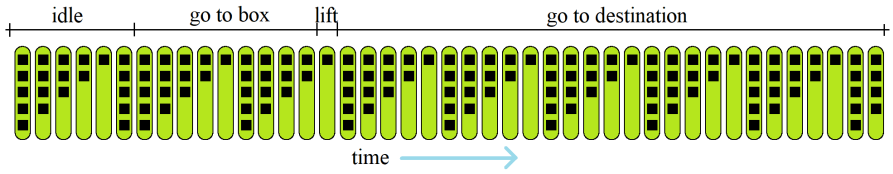

Figure 5: Action Buffer changing over time

2.3.2 Master Slave Mode

2.3.3 Dead Lock

3 Experiment

The domain is shown in Figure (6). Where in the warehouse, there are 49 boxes for small or large size randomly generated. Robots as green are generated on a corner of the warehouse. The goal is to deliver the small boxes to blue zone and large box to red zone. small box can be lifted and carried by a single agent and large box requires 2 agent carrying cooperatively.

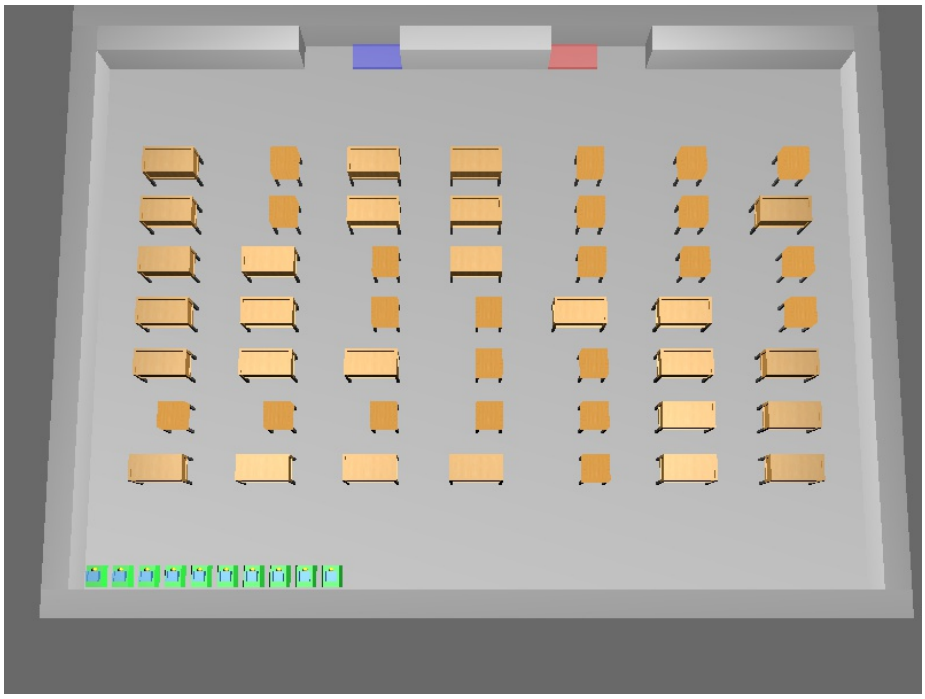

Figure 6: Domain

References