How delegation through intermediate points affects the scalability of the reactive agent system?

Maxim Popov, Asier Serrano Aramburu, Omadbek Meliev

TECHNISCHE UNIVERSITÄT BERLIN

Introduction

Pickup & delivery tasks in the real world usually involve multiple entities (agents) interacting in a common space (environment). For example, logistics providers use hierarchical "Hub and Spoke" model, where most delivery vehicles move packages until certain intermediate distribution centers [1].

This way it is simpler to establish routes and deliver packages to multiple destinations using less agents, since packages can be moved in a more systematic way.

At the same time, reactive agents offer scalable and dynamic behaviour, due to local decision making.

In the present work, we develop a multi-agent system for package distribution in a dynamic grid, and propose delivery delegation using intermediate package points.

Environment

NxM grid, where there are:

- **Package points** (PPs)
  All PPs have unlimited capacity and divide in 3 categories:
  - Starting (SP): only 1 at the center of the grid. Packages periodically appear on it to be delivered.
  - Intermediate (IP): where agents can leave the package, so that others continue the delivery.
  - Ending (EP): packages destination.
- **Agents**: can move vertically and horizontally, one cell at a time, perceive everything in one cell radius. Agents can pick up and deliver packages. They can be placed in SP or IP. Their initial position can be specified, or else automatically determined by a circular priority queue that places them in the intermediate package points closer to the center of the grid. We assume that collisions between agents are not possible.
- **Packages**: They are assigned a random ending package point and a maximum number of iterations to be delivered. They are either at a package point, or with an agent.
  SP also assigns package intermediate point that is closest to its destination.
  Once delivered, they disappear.
- **Obstacles**: where no other entity can be. They appear randomly, stay for i iterations, and disappear. This way we make our environment more dynamic. Obstacles cannot be at the same positions as other entities

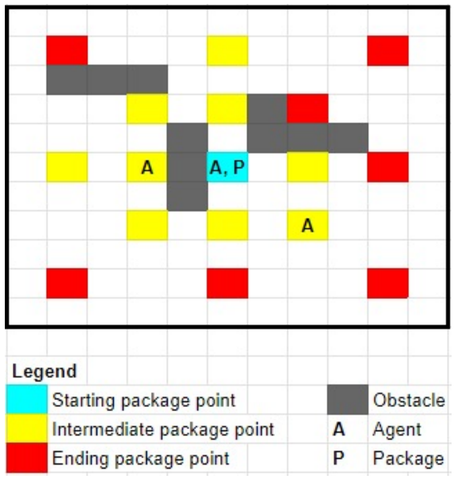

Figure 1. Example of possible starting environment.

Agents Behavior

We use reactive agents that implement simple rules and are able to perform well in dynamic environment.
They have no memory nor specific goals.

Strategies

- **ChainAgent**: Agent waits for package in its origin point. It delivers packages only to a certain type of package point (IP or EP) that is defined from the beginning. This divides all agents in two groups (IP agents and EP agents accordingly). Once delivered, it returns to its origin to wait for more packages.

  This strategy implicitly promotes collaboration, as IP agents pick up packages at SP, leave them at IPs, where they can be picked up and delivered by EP agents.

Path finding algorithms

- Dijkstra
- Pheromones

Experiments Setup

- 1000 iterations per experiment.
- 20 agents with ChainAgent strategy.
- Path finding algorithm is Dijkstra.
- Starting point generates 5 packages every 5 iterations.
- IPs & EPs are placed automatically by sub-rectangles of 8 points. There are more ending points on the outskirts of the grid.
- Agents positions are automatic as described earlier.
- A common random seed 0 is set.
- 3 square grids with side 10, 25 & 50.
- 3-to-1 IP-to-EP ratio. IP/EP: 1/3, 2/6, 4/12.

Results

Figure 2a. Median delivery time using different proportions of IP/EP agents

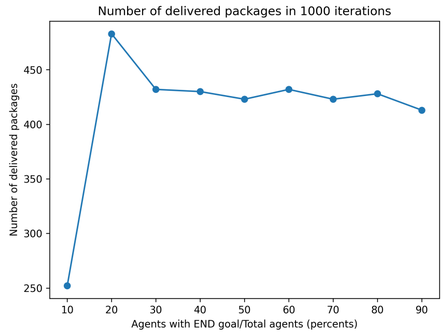

Figure 2b. Number of delivered packages using different proportions of IP/EP agents

Preliminary experiments showed that pheromone path finding performs worse in our environment than Dijkstra algorithm. In our first experiment, we try to determine, which proportion of IP/EP agents provides best performance. We use GRID2 map, while changing the percentage that EP agents take from the overall agent amount.

As seen in Figure 2a, median delivery time rises with amount of EP agents. Meanwhile Figure 2b shows that the overall number of delivered packages is at its highest, when we are at 20% of EP agents (80% IP agents). Possible explanation for that could be that although low amount of EPs reduces the delivery time, this isn't sufficient for delivery, since not all IPs are provided an agent that can pick up packages from them.

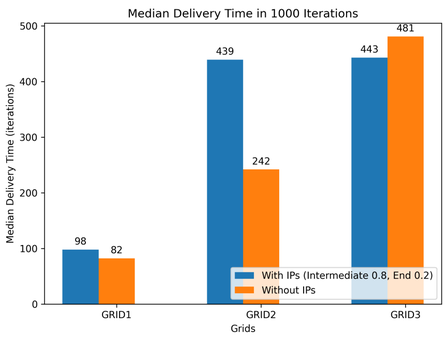

Figure 3. Median Delivery Time with and without Intermediate Points

In the next experiment, we used this proportion of agents on grids of different sizes. Orange bars on the Figure 3 show median delivery time using direct delivery from SPs to EPs, while blue show median time in case when packages are delivered through IPs. Although results for GRID1 and GRID2 show that agents with IPs are performing worse, we can see that in GRID3 the delivery time with IPs is better than without. We can also see that delivery time did not change a lot comparing to GRID2, while grid became twice as big and amount of agents did not change.

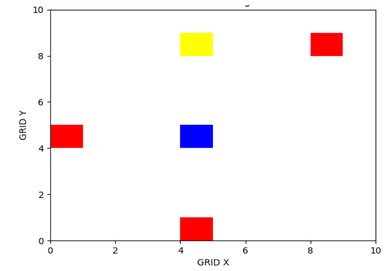

Figure 4a. GRID1 (10×10)

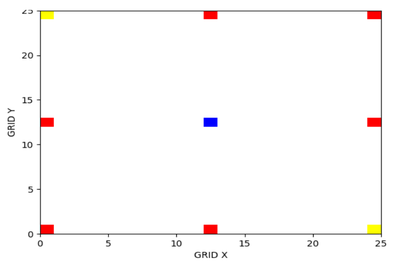

Figure 4b. GRID2 (25×25)

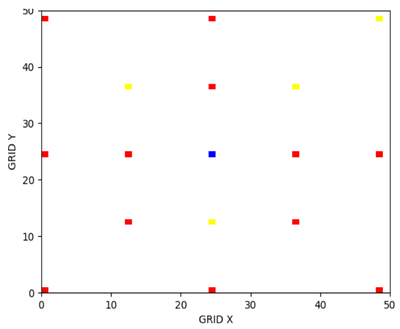

Figure 4b. GRID3 (50×50)

These results can be explained by looking at the PP placement of grids. Random placement of IPs on GRID1 (Figure 4a) resulted in non-optimal location, as the IP is far from most EPs and agents go longer distances to deliver the package. Placement of GRID2 (Figure 4b) is also non-optimal, since IPs are placed at furthest position from SP. GRID3 (Figure 4c) so far, looks as the most feasible placement, as different IPs can be assigned to packages and they are less further to EPs if we compare them with other grids.

Conclusion

- Delegation can help maintain same delivery time in scaling environment, while using same amount of agents.
- However, placement of agents and intermediate points highly impacts performance. Placement strategies may be a subject for further improvement of the model.

Future Work

We identified following possible directions for improving our current work:

- Create placement strategies for different agent groups, as well as intermediate points.
- Introducing different strategies for agent behaviour, such as:
  - **Greedy Agent**: picks up the most optimal package he can perceive and directly delivers it to its ending package point.
    It can use 4 optimality metrics:
    - Closest package to the agent.
    - Closest package to the agent not delayed.
    - Package closer to be delayed.
    - Package closer to destination.

  This strategy does not collaborate. If no destination is set, a random feasible move is done. This may provide better performance in case of bad agent/IP placement.
  - Introducing communication that can help better schedule the delivery and utilize limited amount of agents more efficiently.
  - Agent strategies that take into account crowdedness and resource utilization.

[1] Claes, Rutger, Tom Holvoet, and Jelle Van Gompel. "Coordination in hierarchical pickup and delivery problems using delegate multi-agent systems." In Proceedings of the 4th Workshop on Artificial Transportation Systems and Simulation, pp. 1-7. 2010.