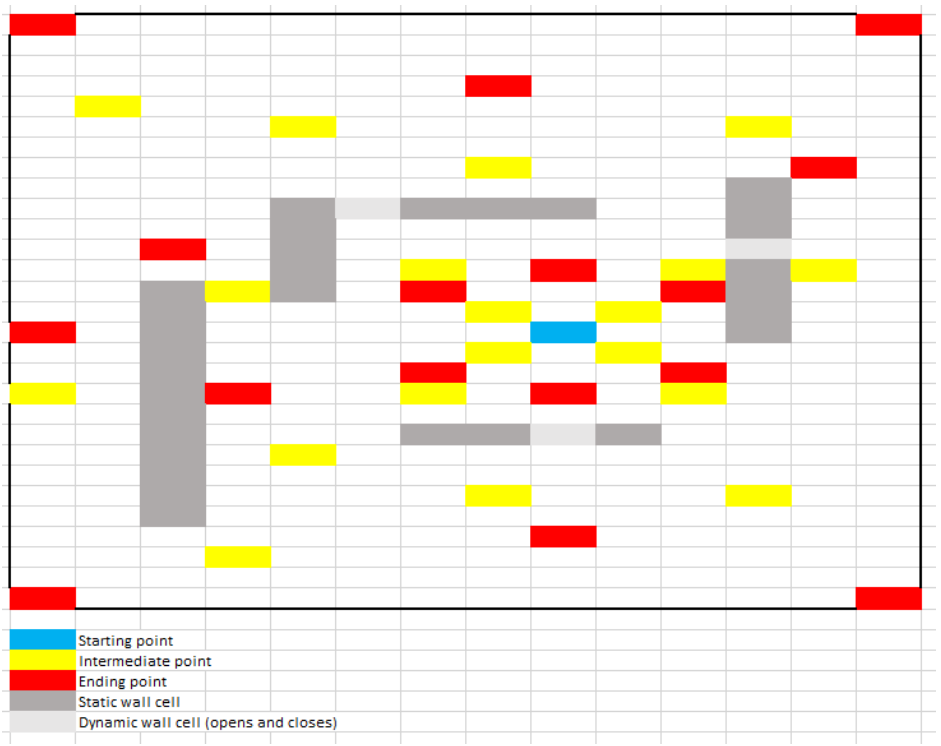
Introduction to Multi-Agent techniques

Deliverable 1 proposal: Reactive Agents

Group G

Environment



A grid of nxn cells, where there are:

- Agents
- Obstacles (walls) with:
    - Static cells: no agent can step there during the whole experiment.
    - Dynamic cells: when they are 'opened', agents can pass through.
- Package points:
    - Starting point: where packages first appear to be picked up. They always appear here, it is like a big logistics center.
    - Middle points: where packages are placed so that other agent at some point picks it up and continues the delivery.
    - Ending points: where packages have to be delivered at the end.
- Packages: they can be at:
    - A package point.
    - An agent's back.

Basic rules & constraints of the environment:

- At each cell, there can only be 1 agent, or 1 obstacle, or 1 package point at the same time.
- Nothing can leave the grid.
- All elements are generated randomly with the same seed.
- An agent can only carry 1 package at a time.

- An agent can not look further than x cells around him (no total vision).

Goal: Agents must deliver as much packages as possible, and as fast as possible. For doing so, we will try that they indirectly collaborate, by tweaking the distribution of each type of agent we have, until we find a somewhat optimal overall configuration.

Classes

Agent

- Parameters
    - ID
    - Position
    - Package
    - Perception
    - Strategy
- Methods
    - step()

Obstacle

- Parameters
    - ID
    - Position
    - Size
    - visibility

Mobility (parent class)

- Manhattan

  Parameters:

    - speed

  Methods:
    - move()
- Non-euclidean (tbd)

Vision

- Parameters:
    - n_cells_around
    - Environment
- Methods:
    - environment_around

Package

- Parameters:
    - ID
    - Position
    - max_iterations_to_deliver
    - iterations_count
    - delivered
- Methods:

    - decrease_time_left

PackagePoint

- Parameters:
    - ID
    - Position
    - packages
    - max_num_of_packages_allowed
- Method:
    - store_package(Package)
    - retrieve_package(Package)
    - distance_to_package_point(PackagePoint, Position)

Grid

- Parameters:
    - Width
    - Length

Environment

- Parameters:
    - Grid
    - Agents
    - Obstacles
    - PackagePoints
    - Packages
    - max_iterations

Logger

- Parameters:
    - dir
    - files_format
- Method:
    - log()

Plotter

- Parameters:
    - dir
    - files_format
- Method:
    - plot()

Experiment

- Parameters:
    - ID
    - Environment
    - Logger
    - Plotter
    - max_iterations

- Methods:
    - run()

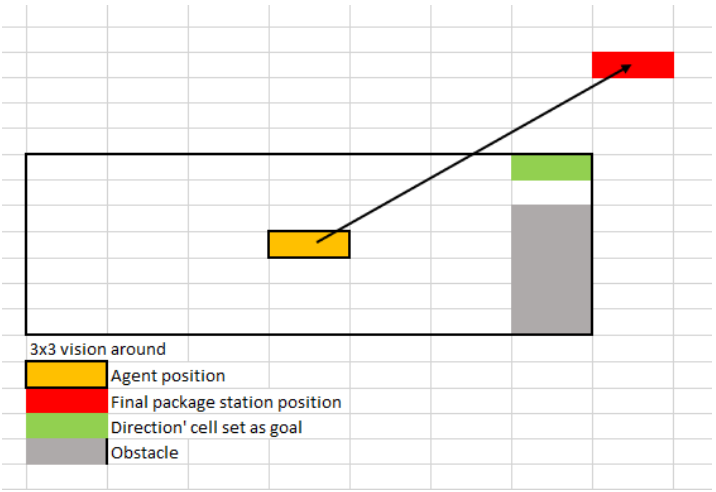
Position

- Parameters
    - X
    - Y

Movements

All strategies share the same procedure in terms of movement:

1. Define a goal, which can be depending on:
    a. The agent is currently carrying a package (has information about the package start, final & time left):
        i. A specific cell the agent sees, such as the position of a package point, where the agent wants to go to according to his strategy.
        ii. A 'direction', which is the closest cell the agent can currently see and is reachable, that is closer to the final package station. See an example:



    b. The agent is currently not carrying a package:
        i. The strategy sets up a goal.
        ii. The strategy does not set up any goal, then define the goal as a randome reachable cell around him.
2. Use Dijkstra to find the fastest path and take a step (unless the decision has been random, in this case there is nothing to optimize).

- How to define the goal cell?

It depends on the agent strategy, and where it is finally allowed to move.

- What are impossible steps?
    - A cell where there is CURRENTLY one of:
        - An obstacle.
        - A package point.
        - A cell out of the grid.
- What if the agent finally cannot move?

Then, the agent will choose its current cell (won't move).

Strategies

- Randomly generated agents:
    - The greedy agent: wants to deliver as fast as possible.
    - The balancing agent: will move packages from one point to another to balance their number.
    - The follower agent: will follow agents with a package if doesn't have a package.
    - The direct agent: does not use the middle points, takes a package and straight only looks for the destination until finds it.
- Strategically positioned agents:
    - By sub squares
    - By triangles

Metrics

- Percentage of packages delivered on time
- Mean delay in percentage of the max iterations to deliver
- Mean path efficiency: compare the initially optimal path of each package, with the final path it follows https://www.mdpi.com/1424-8220/20/11/3118
- Mean effectiveness of collaboration
- Mean number of collisions

Logging

- Agent positions DataFrame:
    - Agent ID
    - Iteration number
    - X
    - Y
- Agent movements DataFrame:
    - Agent ID
    - Iteration number
    - Movements priority queue
    - Collisions found -> (cell, [agents])
    - Final movement position in the priority queue
- Packages pick/drop DataFrame:
    - Package ID
    - Agent ID
    - PackagePoint ID
    - Iteration
    - Action [Pick/Drop]
- Agent carry package DataFrame
    - Package ID
    - Agent ID
    - Iteration

Plots

- Packages
    - Heatmap of the package's movement all over the grid

    - X axes experiments setup, Y axes each metric (X axes sorted by that metric)
    - Interchanges of packages by package point
- Movement
    - Heatmap of the agents movement all over the grid
- Collisions
- Combined
    - Correlation between

<u>Research questions</u>

- How to improve scalability? All towards scalability.

Questions for the professor:

- Is the environment dynamic enough for reactive agents?
- Should we add dynamic obstacles?
- Which metrics should we focus at to answer our scalability question?