Introduction – start on page 1, describe the background information of the problem and briefly include all requirements given by the customer (instructor). The latter also includes new/additional requirements.

The file must be verified. Each algorithm had different constrains.

Users could make agents run at one step or multiple steps at a time. Before new/additional requirements added, when an agent chose a target, this target would be deleted from target list. The updated algorithm constrained3 and constrained4 would not delete a target unless it was visited by an agent. The number of agents in a region in Constrained-3 algorithm can at most be [n/3] where n is the number of open spaces in that region. The number of agents in a region in Constrained-4 algorithm can at most be [n/4] where n is the number of open spaces in that region. The agents in each region should be placed in the end nodes. The agent who had visited less nodes had priority to choose target.

• Implementation strategies

1. Implementation of read file function

The read file function read each line of the file and grasped the key words like “Size”, “Agents”, “Agent”. Regular Expression was used to match the numbers showed behind those key words. So, all coordinates in one region must in one line. An ‘environment’ variable was used to store the all the information read from the file. Environment had three attributes, size, regions and agents. ‘Regions’ was an object which had id and open spaces attributes. ‘Agents’ was an object which had id, position, and region attributes. This was the structure of environment.

1. Implementation of free-form algorithm

Free form algorithm chose the first node from the target list. When the target was chosen, it would be delete immediately. The target list is generated according to the environment which is read from the input file. The environment information included regions. This project iterated the all regions and add all open spaces in each region to the target list. At the beginning of this project I used DFS algorithm to find the shortest path. Then I realized that the algorithm was not good enough. The Node NPM has a package named ‘PathFinding’ module. This module is very good at finding a path. Instead of using my DFS algorithm, I chose the third-party package and using ‘A-Star’ to find the shortest path. In order to separate the front-end and backend code, the backend algorithm get the paths for each agent and all the target list. The algorithm was basically found a target, agents moved towards the target and found another one. Here are more details about how it works. Firstly, each agent chose his target and got a path. The path was stored in a temp local variable. Every time removed a node from each agent temp local variable path, which meant move each agent one by one then stored the removed node as a part of agent path. When the temp local path variable was empty, chose another target and generated another path, storied the path variable until there was no target for this agent. By doing this, all the small paths added together became one path for an agent. The target list was recorded every time when agents moved one step.

1. Implementation of constrained-3 algorithm

Constrained-3 algorithm chose the farthest target in the target list. The farthest target was found by shortest path finding algorithm. It iterated the target list one by one and calculated each distance between the current position and target. When a target was found, it would not be deleted. The node in target would be deleted when an agent visited the target. If an agent moved when there was more than one choice to move to the next step, it would choose the first target in the target list. The way to get an agent path, all targets, and all target lists were the same as free form algorithm.

1. Implementation of constrained-4 algorithm

Constrained-4 algorithm was like free-form but not deleted the target when an agent chose his target. It calculated the steps an agent had moved, whoever moved the less steps had priority to choose the target. The way to get an agent path, all targets, and all target lists were the same as free form algorithm.

1. Implementation of verify file

Once the file was loaded, it would verify the file immediately. It verified the size, the agent id was duplicated or not, region id was duplicated or not, coordinates of open spaces was valid or not, agent in his region or not, the number agents in each region etc. Go through the environment variable to check the file was valid or not.

– describe how your team implemented the project; explain your design choices. Examples for design choices include GUI design, how an agent moves when there is more than one choice to move to the next step and so on.

• Technology and tools – those you used in implementation. Do not just name them but include at least one or two sentences for each one of them.

Node server: Node has NPM (Node Package Manager). NPM manages a lot of packages. Packages like libraries which are convenient to use.

• Challenges – explain what challenges your team faced in implementing this project; include technical as well as personnel challenges.

At the beginning, the implementation of DFS algorithm using JavaScript was a difficult. I was not familiar with JavaScript. The updated algorithm was using third-party library which was very easy to use. The logic of generating all the target lists and storing each agent’s path were difficult. Figuring a way to story those data and make sure each target list, target, and agent position match were tricky. Also, it was difficult to testing. Because the target list was too big, the path of each agent was too long, plus multiple agents moved at the same time. Graph view was needed to test the algorithm.

• Learning – what you learned from this project.

Finish you job perfectly, professionally and on time. Don’t expect others clean it up for you. If you don’t finish your part perfectly, you will affect this project and cause problems to other team members. Attitude matters a lot. I have also learned how to cooperate with team members. Everyone is different, for example, someone like to finish job early and others might wait until the last minute, someone may not like the way other team member talks when conflicts happened. Those problems are challenges when we work together. Always keep in mind that we are a team. When a team member has some problems other team members should help him/her. Put more faith on your team mates, believe they can get their work done.