**A Tool to Visualize Multi-Agent Patrolling**

A project report submitted for

CS 744 - Management Issues in Software Engineering

**Team: Giant**

**Team member: Dezheng Wang, Weikang Wang, Ying Jiang**

**Instructor: Dr. Kasi Periyasamy**

May 3, 2017

# **Introduction**

The goal of the project was to develop a web-based application associated a web server that helps visualizing multi-agent patrolling in different environments. As given in project description provided by instructor, the environment consists of a rectangular array of squares; Each square represents either an open space or an obstacle. The open spaces are somewhat connected so that one can visit from one open space to the other open spaces.

The application should use a block view and a graph view to display the environment and the agents at any time. The block view should display the whole environment and all agents in that environment. However, the graph view only need to display one region and related information. In addition, patrol algorithm should be developed to visualize agents patrolling. This document would describe all requirements from different perspectives. Because requirements had been changed during implementation, it divided requirements into two stages.

Stage one: Initial requirements to changes on requirements

* Block view: Environment contained open spaces close spaces and agents. Open spaces contiguous to each other form a region. Every agent show in their start position in different color with its ID. If run for one or more steps, line will show in the agent's color as agent's path. If users want to specify a single agent's path. They can click the node list by ID on the left side of environment to make the path show on the top layer. Click on squares list by the side with region ID can make the page change to Graphical view.
* Graphical view: Each region in the environment should have graph view; Nodes of a graph represented the open spaces in that region; If two open spaces were adjacent to each other, they should be connected through an edge; The content of a node indicated the number of agents in that node; Double-click on a node should display details of agents in that node (etc. node id); The position of a node should show by the side of the node; Color should be used to indicate whether a node was visited at least by one agent; There should have two buttons-one to execute the algorithm step by step and the other to execute the algorithm for a fixed numbers of steps.
* Algorithm: Users could make agents run at one step or multiple steps at a time. When an agent chose a target, this target would be deleted from target list. The number of agents in each region should be [1-n/2]. The id of each agent was unique but not had to start from one. Multiple agents can be placed at the same position.
* File validation: For free-form file validation, the constrain was that the number of agents in each region should be [1-n/2]. There will be at least one open space in the environment. The open spaces in one region are all connected so that it is possible to visit from any open space to any other open space within the same region. Agents are always placed in open spaces. Those three constrains were basic constrain for the environment. No matter what algorithm a user select, those constrains need to be obeyed.

Stage two: After changes on requirements

* Block view: Not too much change in block view, just make the style looks better.
* Graphical view: Target list in a region should be included in the graph. What’s more, the graph should show the current target selected by each agent in that region.
* Save and search run: Users should be able to save each run after determination of the algorithm. Following run information should be saved: Regions; Coordinates of the region; The target list for the region; If the target list was changed at step N, that information should be saved; Agents; The trace of each agent’s movement; Number of steps to complete this run; As for searching run history, users should have multiple choices. Users could search run information based on one or more of the following information: date, time, size of the environment, number of regions, and number of steps for completion. After run history were retrieved, all those run statistics mentioned in saving run should display.
* Algorithm: 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.
* File validation: More constrains were added to Constrained-3 and Constrained-4 algorithm. In constrained-3 algorithm, the number of agents in a region can at most be [n/3]. In constrained-4 algorithm, the number of agents can at most be at [n/4]. The position of agent was also constrained to the end nodes. If there was not end node in a region, the agent can be placed at any node.

# **Implementation strategies**

This section will talk about design choices and how the team implemented the project. It would be described in several aspects.

* Block view: Weikang take the charge of front side. Because the system is mainly about the front side, Dezheng will take the graphic view part to relieve my work. In the beginning, we decide that user can initial environment by hand. So Weikang take the charge to develop a function for user to draw environment. Even though we dropped this function later. At the same time, algorithm development and back side development also were in process. First, Weikang decide to fill the open space with another color to mark the paths of agents already visited. But after second demo, Weikang noticed that a group mark the path with colored line. So Weikang changed design and mark the path with colored line. Next, Weikang moved to deal with crossed lines from different agents. After discussing with all group member, we decide to mark agent nodes besides the environment which if clicked, the path of that agent will show on the top. After Ying and Dezheng complete part of their work, Weikang provide the interface for changing from block view to graphic view and receive and show run results get from Ying step by step.
* Graphical view: For graphical view, the main work is to build user interface for those nodes, edges and target list. To draw a graphical view, some run statistics were necessary. The first parameter was region information which included region id, coordinates of open spaces, agent information. Drawing graphics functions were performed by canvas which had several methods for drawing paths, circles and texts. The first step was to draw nodes(circles) to represent open spaces in that region. The coordinate should also be shown by the side of each node. The node would use number to represent agents in it. If multiple agents were in one node, the number was total count of agents. As for a double-click on a node, it would be implemented by using JQuery. It could get actual coordinate of that node and then display agents’ information in that node. Then the application would identify any two vertical or horizontal adjacent open spaces and drawing lines between them. Finally, the application could get static graphical view based on the first parameter. In addition to run algorithm in graphical view, agents’ paths and total steps were needed. If users ran five steps, agents had moved to corresponding positions. The graphical view would get current coordinates of each agent and then draw the graph again. To show visited open spaces, the graphical view would get all coordinates of visited open spaces and changed the color of those open spaces. Target list information and each agent’s target list were needed to show target list and current target. The graphical view would show target list and each agent’s target based on total steps. For example, if a user had run seven steps, the graphical view would get target list and each agent’s target at seventh step. Then it would show those statistics.
* Save and search run: The saving information and save button would only occur when the number of total steps was larger than the number of longest paths in agents’ paths. If users selected to save a run, the application would collect statistics which were mentioned in requirements firstly. Then the application would use JSON method to convert JavaScript values to JSON strings. After all values were wrapped, it would use JQuery method (ajax) to send an asynchronous HTTP request to server to save run.

If users want to search run information, they could select one or more of following information: run id, date, time, size of the environment, number of regions, and number of steps for completion. Time field had used ‘bootstrap-material-datetimepicker’ which was a third-party library. After getting searching options, the application would use JQuery method (ajax) to send an asynchronous HTTP request to server to get corresponding information. If return value was null, it would give error message. If return value was not null, the application would deal with return value and show it in appropriate way. User interface of showing return value had used a third-party framework whose name was ‘materialize’.

* Server side: The team was required to build a server on remote server machine. The first step was to connect to the server. Because the team planned to build HTTP-based applications, Node.js would be used to build the server. So, the remote machine need to download Node.js [3] and install it. Npm [4] was needed because the team used JavaScript as main programming language and Npm was package manager for JavaScript. The server also need to download MySQL because the team decided to use MySQL as database. After all applications had been downloaded and installed, it was time to construct the server. Firstly, the server should be able to receive requests from front-end. Secondly, it should divide request into different categories based on types of requests and contents (URLs) of requests. Thirdly, creating methods for each kind of request. Fourthly, generating different SQL statements according to each method. Fifthly, connecting to database and using corresponding SQL statements to add something or get something and then get return value. Finally, sending return value to front-end. All those steps had appropriate error message if something wrong occurred.
* Read file: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.
* File validation:

Free-form:

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.

Constrained-3:

It had all the constrains that in free-form but the number of agents in one region can at most be [n/3]. Just changing the condition would be enough for constrained-3.

Constrained-4:

It had all the constrains that in free-form but the number of agents in one region can at most be [n/4]. And agents must be initially placed at the end nodes of the region. In other words, there is only one way for an agent to move from its initial position. If there is no end position in the region (rectangular or square shaped regions), the agent can be placed anywhere in the region.

* Algorithm:

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.

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. 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.

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.

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.

# **Technology and tools**

In this section, each person would describe technology and tools used in implementation. All team members used GitHub to manage the project.

* Dezheng Wang: The IDE used for this project was Visual Studio Code and the main programming language used to design graphical view was JavaScript. To manage packages in JavaScript, all team members chose Npm. The HTML <canvas> element was only a container for graphics. The application used JavaScript to draw the graphics. I had added 'SweetAlert' into package and used it which was a beautiful replacement for JavaScript's 'Alert'. For showing searching information, the main third-party framework was ‘materialize’. The biggest advantage of using materialize was that it would only show key information if users did not click it. Plenty of information would be presented when users clicked it. For the server side, Node.js was used to build the server because JavaScript was the main programming language of client side. Because the team decide to use MySQL as server database, package named 'mysql' should be used to connect to MySQL. MySQL workbench was also needed to manage MySQL.
* Weikang Wang: The IDE used for this project is Visual Studio Code and the programming language used to develop block view is JavaScript. Raphael is the library Weikang used to draw Block View.
* Ying Jiang: Visual Studio was used in this project. Node has NPM (Node Package Manager). NPM manages a lot of packages. Packages like libraries which are convenient to use. Express framework was used. The data sent from front-end to the back-end via Ajax call.

# **Challenges**

This section would explain what challenges the team had met during implementation. Then each person would indicate their own challenges.

* The team’s challenges: The biggest challenge of the team was that different team member had different knowledge levels. It was hard for the team to decide what tools and which programming language to use in the development. It did take several hours for the team to decide to use JavaScript as main programming language and use MySQL as server database. The division of this project was another big problem because each member was familiar with different knowledge fields. Another big challenge was the first integration between each member’s part. The team did not make some universal data structures very clear and it brought lots of troubles when the team had its first integration.
* Dezheng Wang: When I was developing the graphical view, the biggest challenge was how to draw those nodes, edges and draw lines between them. At the very beginning, I planned to use d3 to draw graphics. D3 was a JavaScript library visualizing data with HTML, SVG and CSS. I had tried learning how to use it but it was a little hard for me. Finally, I chose a simple way, using canvas to draw graphics. When I was developing the user interface of searching run information, I planned to use table to use run data. However, a table could not show all data and the table looked strange. I asked help from my classmates and they recommended to look at materialize. I took their advice and found that it was quite easy to use materialize and the user interface looked very nice. Another challenge occurred when I was installing MySQL workbench on remote server machine. It always asked me to install two kinds of frameworks before installing MySQL workbench. I had checked the system and was sure that both had been installed. It took me several hours to figure that out. The problem was that one of the installed framework's version was too low to meet the requirement.
* Weikang Wang: In general, there is no technology challenge for me because I have already written some web application. One problem for me is choosing a proper library for drawing block view and get used to it. Another one is how to integrate my group members' part with the main page because I take the charge of designing main page and I haven't work with Dezheng and Ying before.
* Ying Jiang: 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**

This section was related to each person’s own experience. So, everyone in the team would make a conclusion on what he or she had learned from this project.

* Dezheng Wang: During the process of developing this application, I had learned how to use agile method to develop a product. Once the team decided what kind of tools and language would be used for development, each team member should learn some knowledge to keep in the same level (a little bit higher or lower is OK). Agile development required the team to held several face-to-face meetings to find out user stories and sprints for each cycle (etc. one week or two weeks). Each user story was divided into small sprint tasks and it usually took one or two days to complete each sprint task. Some sprint tasks should be tested. It was better to have meeting logs to record each member's contribution and task. I had also learned how to act better while I was working with another team member. When conflicts occurred, the best way to use your words to explain it to others instead of ignoring others. Face-to-face meeting was a good solution to communicate with others to solve problems. The last thing was to finish personal work on time in case of having influence on the whole team.
* Weikang Wang: Develop project with Scrum is absolute a new process for me. Before this, I never tried developing a project with so many detailed documents. Through the Scrum, I know a lot about the Agile Methods and found that these documents really help our work close to the plan and always care about the timelines. Anyone's contribution will affect the efficiency of whole project. Now I understand more about the importance of documents. Another one, since we are a team, allocate work according to single member's ability is a good way to increase the speed of development. Only with all the members working close to each other, project can be completed on time perfectly.
* Ying Jiang: The procedure of developing a project using Scrum was clear to me. I got a better understanding of terms in Scrum development. I had learned how to use document to record the process of a project. Other thing was how to get along well with team mates. Finish you job perfectly, professionally and on time. Don’t expect others to clean it up for you. If you didn’t finish your part perfectly, you would affect this project and cause problems to other 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 were a team. When a team member had some problems, other team members should help him/her. Put more faith on your team mates, believe they can get their work done perfectly. Some coding experience also gained from this project. I had a better understanding of web framework and development a web application.

# **References**

There were no references for this document.