AI Cooperative Path Finding

Spring 2014

Brett Henne

April 20, 2014

Table of Contents

[Table of Figures 3](#_Toc385612242)

[Summary of System Functionality 4](#_Toc385612243)

[Description of AI Algorithms and Methods 6](#_Toc385612244)

[Dijkstra's Algorithm 6](#_Toc385612245)

[A\* Algorithm 7](#_Toc385612246)

[Distance Methods 8](#_Toc385612247)

[User Interface Descriptions 9](#_Toc385612248)

[Section: AI Parameters 9](#_Toc385612249)

[Section : Map Generator 9](#_Toc385612250)

[Section : Simulation Preferences 10](#_Toc385612251)

[Section: Control 10](#_Toc385612252)

[Section: Help Status 11](#_Toc385612253)

[Section: Map 12](#_Toc385612254)

[Section : Map Edit 13](#_Toc385612255)

[Section: Data Log 14](#_Toc385612256)

[System Evaluation 16](#_Toc385612257)

[AI Method Comparisons 16](#_Toc385612258)

[Evaluation Criteria 16](#_Toc385612259)

[Test Cases 17](#_Toc385612260)

[Test Case #1 17](#_Toc385612261)

# Table of Figures

[Figure 1 - UI Screenshot 3](#_Toc385611498)

[Figure 2 - AI Algorithms & Heuristics Block Diagram 4](#_Toc385611499)

[Figure 3 - Dijksta's Algorithm 5](#_Toc385611500)

[Figure 4 - A\* Algorithm 6](#_Toc385611501)

[Figure 5 - Distance Methods 7](#_Toc385611502)

[Figure 6 - AI Parameters Screenshot 8](#_Toc385611503)

[Figure 7 - Map Generator Screenshot 8](#_Toc385611504)

[Figure 8 - Simulation Preferences 9](#_Toc385611505)

[Figure 9 - Control Screenshot 9](#_Toc385611506)

[Figure 10 - Help Status Screenshot 10](#_Toc385611507)

[Figure 11 - Map Area Screenshot 11](#_Toc385611508)

[Figure 12 - Map Node Editor Screenshot 12](#_Toc385611509)

[Figure 13 - Data Log Screenshot 13](#_Toc385611510)

# Summary of System Functionality

The prototype is written in C# and was designed using the MVC design pattern. Each agent navigates the map on a separate background thread. All of the AI algorithms have been implemented using an abstract class that provides implementation for common methods and abstract definitions for required methods. A screenshot of the UI is shown below in figure 1. The UI can be divided into the following sections:

* AI Parameters
* Map Generator
* Simulation Preferences
* Control
* Help
* Data Log
* Map
* Map Node Editor
* Help Status Indicator

Each section is described below. The software is able to generate random and empty maps, edit walkable/non walkable nodes, edit agent start/finish locations, and draw multiple agents path finding through the map simultaneously . Agents also have the ability to cooperative by sharing map information when they are within the field of view of one another. The agents navigate through the map using 1 of 2 path finding algorithms, A\* or Dijkstra. These AI methods were chosen because A\* uses a heuristic to determine the best node in a set and Dijkstra does not. The AI system incorporates four heuristic methods for determining distance to nodes:

* Euclidean Squared
* Euclidean
* Manhattan
* Chebyshev

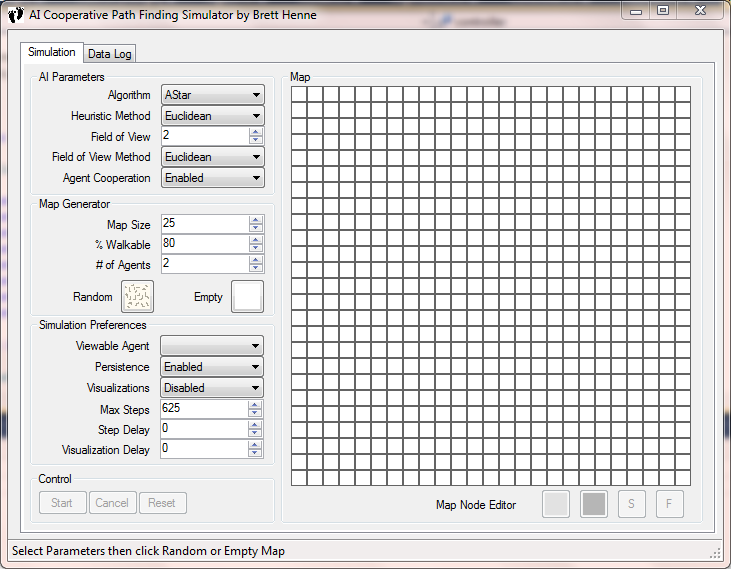


Figure - UI Screenshot

Figure - AI Algorithms & Heuristics Block Diagram

## Description of AI Algorithms and Methods

### Dijkstra's Algorithm

This algorithm maintains a visited an unvisited list to keep track of the nodes in the agent's field of view. All node values are set to infinity except the starting node. The distances of the unvisited neighbors are calculated and added to the current nodes cost value. During this step the parent node is updated for use in back tracking when the algorithm has completed. If the calculated value is less than the value stored in the node the value is updated. Once all nodes have been evaluated the algorithm selects the node with the lowest value from the unvisited list and places that node in the visited list. The algorithm is repeated until the target node is found or if the next node to be evaluated has a node value of infinity, in which case the algorithm terminates. The flowchart of Dijksta's algorithm is shown below in Figure 2.

Figure - Dijksta's Algorithm

### A\* Algorithm

This algorithm is an extension of Dijkstra’s algorithm with a heuristic approach.  The algorithm maintains an open and closes list to keep track of the nodes.  The open list maintains a list of nodes to be evaluated that may or may not be on the shortest path.  After all adjacent nodes to a given node have been evaluated the node is moved to the closed list.  The next node is chosen by calculating a path score.  The path score is a combination of the movement cost to move from the starting node to the selected node plus the estimated cost to move from the selected node to the destination node.  The second part of this formula is the heuristic.

Figure - A\* Algorithm

## Distance Methods

The system incorporates four different methods for determining distance between nodes. The distance methods are used in the field of view method as well as the heuristic method for the A\* algorithm. The distance calculations are illustrated below in Figure 5.

Figure - Distance Methods

## User Interface Descriptions

### Section: AI Parameters

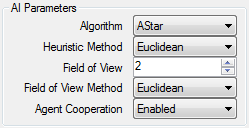
****

Figure - AI Parameters Screenshot

This section gives the user the ability to select which path finding algorithm to use, the Heuristic method used (if applicable), the field of view of the agent, the field of view method used, and the ability to enable/disable agent cooperation. The field of view allows the user to define how far the agent can see. This affects how many nodes are in the agents map and how many nodes the algorithms use in determining which steps to take. There are two AI algorithms implemented in the system, A Star and Dijkstra's algorithm, of which the latter does not use the heuristic. The A\* Algorithm has the ability to select the distance method used when calculating the heuristic.

### Section : Map Generator

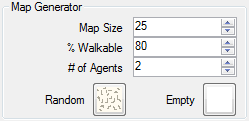
****

Figure - Map Generator Screenshot

This sections gives the user the ability to generate random or empty maps of any size within reason. The agent(s) start and finish locations are randomly chosen among the set for both map types. The size of the map, the percentage of the map that is walkable, and the number of agents are all variables that can be modified. The software limits the number of agents currently to 10.

### Section : Simulation Preferences

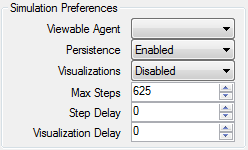
****

Figure - Simulation Preferences

This section is where various preferences can be set.

* Viewable Agent - this allows all agents to be viewed or just one.
* Persistence - this allows the path to remain colored or cleared as the agents walk through the map.
* Visualizations - this illustrates to the user how the nodes are being evaluated.
* Max Steps - this sets the maximum steps the agent can take before termination. This is needed due to scenarios of limited agent visibility. In these case agents can sometimes get trapped into areas on the map in which the algorithm selects between 2 alternating nodes indefinitely.
* Step Delay - this allows the user to modify the pace of the simulation, allowing for slower or faster runs based on needs.
* Visualization Delay - this allows the user to modify the pace of the visualization, allowing for slower or faster illustrations.

### Section: Control

****

Figure - Control Screenshot

This section is used to control the start and stop of the simulation and to reset the fog of war visibility of the map. It contains three buttons:

* Start - this will start the current simulation based on the current AI preferences and latest generated map. If Map Generator settings have been modified without clicking the Random or Empty button, these settings will be discarded.
* Cancel - this will asynchronously stop all agent threads.
* Reset - this will reset the map to the initial condition.

### Section: Help Status

****

Figure - Help Status Screenshot

After having my wife try to use the software I quickly realized that I needed to add a Help section that guides the user. This section will display help information during various phases of run-time.

### Section: Map

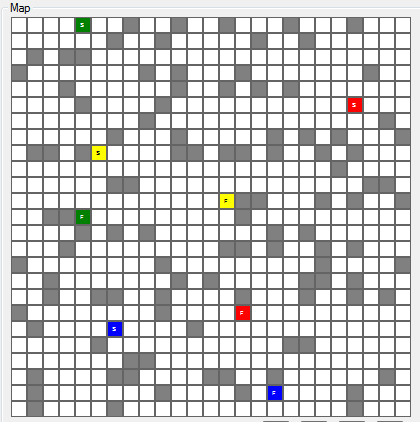


Figure - Map Area Screenshot

This is the map that all agents modify in order to show their start, finish, & current locations. The walkable nodes are white, and the non walkable nodes are gray. The colored nodes are agent start and finish locations, post-fixed with S and F respectively. It has a hidden feature that will display a tooltip of the nodes coordinates when the mouse is hovered over a node. This feature has been beneficial during various debugging phases of the prototype.

### Section : Map Edit

****

Figure - Map Node Editor Screenshot

This sections gives the user the ability to modify the walkable/non-walkable nodes and the start and finish nodes for each agent. The walkable node is the white button and the non-walkable node is the gray button. The agent start node is the button labeled with "S" and the agent finish node is the button labeled "F". Multiple clicks of these buttons will cycle through the available agents which are distinguished by color. The state of the edit feature is based on the last button clicked, therefore if the last button was the yellow "S" above then every click in the map will move the yellow agents start node every click. If the last button clicked was the white walkable button then every click in the map will modify the node to walkable. Nodes that are currently selected to be an agent's start or finish node cannot be modified to walkable/non-walkable. The walkable/non-walkable nodes can be clicked individually or by dragging the mouse across the map.

### Section: Data Log

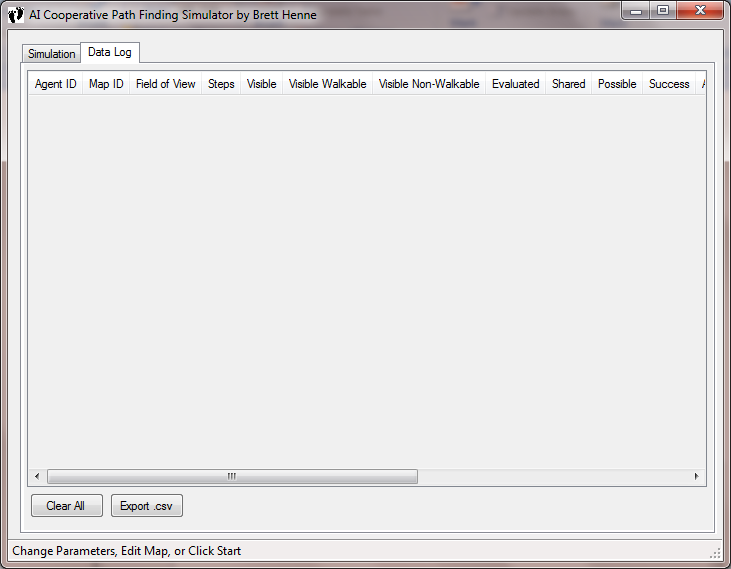
****

Figure - Data Log Screenshot

This section is where all the simulation results will be viewed by the user. This section contains two buttons.

* Clear All - This will clear all the contents
* Export .csv - This will open a dialog allowing the user to save the current table data to a CSV file.

The following results are being displayed in the table per agent:

* Agent ID
* Map ID
* Field of View
* Steps
* Visible
* Visible Walkable
* Visible Non-Walkable
* Evaluated
* Shared
* Possible
* Success
* Algorithm
* Heuristic Method
* Field of View
* Field of View Method
* Cooperation
* Map Size
* % Walkable
* Agents

# System Evaluation

## AI Method Comparisons

Each AI method will be evaluated over a variety of random maps to include the following variations:

* With and without agent sharing.
* Various field of views
* Various heuristic methods

## Evaluation Criteria

* Algorithm used
* Heuristic used (if any)
* Agent Field of view
* Number of steps taken
* Destination reached successfully
* Number of nodes evaluated to reach destination
* Number of agent shared nodes evaluated
* Number of self discovered nodes in agents map
* Number of agent shared nodes in agents map

## Test Cases

### Test Case #1

Purpose: To evaluate performance of A\* & Dijkstra with zero non-walkable nodes and limited agent visibility of 2.

|  |  |
| --- | --- |
|  |  |
|  |  |

Results: Both algorithms found a path with 16 steps, however A\* performed 128 node evaluations compared to 606 node evaluations using Dijkstra.