Technical University of Denmark

02285 Artificial Intelligence and Multi-agent Systems

Mandatory Assignment 1

Authors:

Andreas Hallberg Kjeldsen
s092638@student.dtu.dk

Morten Chabert Eskesen
s133304@student.dtu.dk

Peter Carlslund
s113998@student.dtu.dk

March 11, 2014

Andreas Kjeldsen (s092638)
Morten Eskesen (s133304)
Peter Carlslund (s113998)

Introduction

This assignment is about path finding in a randomly generated environment. The assignment is an introduction to a larger project with multi-agents.

Problem Analysis

Being able to find paths in a known environment is trivial. However if an agent is dropped into an unknown environment you have to map the environment first in order to be able to find paths. The most interesting part of this problem is exactly that.

There are multiple ways of mapping an unexplored environment, the problem is how do you do it efficiently. There are certain limitations to the mapping process:

- When moving from one spot to another, we do not know what direction we went.
- We do not know how the spots are located in relation to other spots, just that two spots have a path between them.
- We can only see territory that are at most two 'spots' away.

While mapping we should also remember how cumbersome it was to travel from one spot to another and also remember if the spots had valuable resources.

To overcome the challenge of mapping the unexplored environment with the limitations stated above, we need to solve four subproblems: How to map the environment, decide in which order we should gather information, how to plan our actions and how to plan a route from one spot to another.

Solution

Mapping

We start by creating an empty graph, then we add the currently visible vertices and edges to it. We use breadth first search for figuring out which vertex to visit next, this is because we disregard the edge weight and just want to find the closest possible unexplored vertex. Whenever we reach a vertex, we add any new vertices and edges that are now visible. We continue this way until the whole environment has been mapped, which we claim it has been whenever we do not know of any unvisited vertices.

Gathering Information

Each vertex has a value and each edge has a cost. These values are not known simply by being able to see the vertex or edge. The vertices have to be probed and the edges have to be surveyed. Probing and surveying are separate actions. When we reach a vertex that we haven't probed, we probe it. When positioned at a vertex that has one or more edge that are unsurveyed, we survey them.

Action Planning

Our agent supports the following actions: `Recharge, Probe, Survey, GoTo, Skip`. The `Probe` and `Survey` actions are used for gathering information. The `GoTo` action is used for moving the agent to another vertex. The `Recharge` action is used when we're low on energy. Our agent consumes energy when performing actions, therefore it's important to keep track of how much energy is left and recharge when required. We claim that a recharge is necessary when there are $\frac{1}{3}$ energy left. We also make the agent recharge fully, meaning more than one recharge action might be required. We only use the `Skip` action as a fallback action, meaning it will only be chosen if no other action is chosen and the agent is fully charged.

Which action to be chosen is checked for in the following hierarchical order: `Recharge, Probe, Survey, GoTo, Skip`. This ensures that our agent gets recharged and that it does not move to another vertex without first gathering all possible information from the vertex it is currently positioned at.

Path Planning

While mapping, our agent should be able to take instructions about where to go. When receiving an instruction, the agent uses Dijkstra's algorithm to plan the path to the goal vertex. We choose Dijkstra's algorithm because we only have the edge weights and therefore couldn't provide a better heuristic than using the edge weights.

While moving towards the goal vertex, new vertices and edges are still probed and surveyed. If the goal node has not been seen, the agent keep on mapping, but notifies when the goal node has been found.

Results

Our agent is able to properly map an unknown environment, though as of right now we cannot make any guarantees regarding the time it takes to map the environment. When the environment has been mapped, our agent will recharge fully and then stand still until further instructions are send to it, this might not be the perfect way to behave, but we can disregard the opposing team for the moment and therefore an idle agent is okay.

Our agent is also able to plan a path to a specific node and follow the path afterwards. The heuristic used could possibly be improved by somehow transforming the graph into a real map with coordinates, that would allow us to use metric distances between vertices in our heuristics.

We've been experiencing some unusual behavior when running our agent. Sometimes our agent would try to go to the same vertex it was already located at, this is because even though it's first `GoTo` action was successful, the server does not indicate so in the response. We've determined that the server responses are one turn behind. We've been notified that we're not the only group experiencing this unusual behavior.