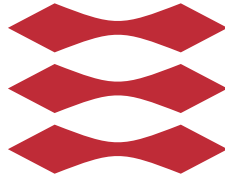# DTU

Technical University of Denmark

02285 Artificial Intelligence and Multi-agent Systems

Mandatory Assignment 3

Authors:

Andreas Hallberg Kjeldsen
s092638@student.dtu.dk

Morten Chabert Eskesen
s133304@student.dtu.dk

Peter Carlslund
s113998@student.dtu.dk

June 2, 2014

June 2, 2014

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

Artificial Intelligence and Multi-agent systems

Contents

June 2, 2014

Andreas Kjeldsen (s092638)
Morten Eskesen (s133304)
Artificial Intelligence and Multi-agent systems
Peter Carlslund (s113998)

1 Introduction

Andreas &
Morten

This project was produced while taking the course Artificial Intelligence and Multi-agent systems at the Technical University of Denmark. The project had to be done in groups of 3 to 6 people. For the development of the project, there were 3 iterations, this is the report detailing the third and final iteration. Each section and subsection is annotated with whom contributed to the writing of the section. Sections regarding implementation specific areas are written by the persons who wrote the actual implementation.

1.1 Scenario

Andreas &
Morten

There are 28 agents with different roles. They are dropped into an unknown environment. The agents have properties (health, max health, energy, max energy), a visibility range and role defining which specific actions they can do. The environment is a graph consisting of vertices and edges where vertices have a certain value and edges cost a certain amount of energy to traverse. Lastly there is the notion of a score which is the accumulated sum of the zone score for a team during the game.

1.2 Problem Analysis

Andreas &
Morten

The scenario of the game introduces a set of problems to overcome.

1.2.1 Gathering Knowledge

Andreas &
Morten

The agents have no prior knowledge of the graph, no knowledge about where the other agents are in relation to each other, nor where the opponent agents are. The agents must be able to share their knowledge and keep track of their opponents.

1.2.2 Working Together

Andreas &
Morten

The agents might wish to perform the exact same action, this is not beneficial, therefore the agents should be able to work out whom should do what. Preferably in such a way that the agent able to perform the action with the probable best outcome, is chosen to perform it.

1.2.3 Scoring

Andreas &
Morten

The agents must coordinate their actions in accordance with an overall strategy that works towards achieving a good score.

1.2.4 Agent Strategy

Andreas &
Morten

A strategy for the agents must be defined. How aggressive should the Saboteur agent be. When should the agents request to be repaired by the Repairer agent. For how long should the agents focus on gathering knowledge and when should they focus on achieving a good score.

June 2, 2014

Andreas Kjeldsen (s092638)
Morten Eskesen (s133304)
Artificial Intelligence and Multi-agent systems
Peter Carlslund (s113998)

2 Environment

Andreas

The environment is randomly generated, that means assumptions about the environment should not be made. Further the agents are placed at random. The agents are allowed to communicate and share their knowledge.

2.1 Map

Andreas

The map of the environment is being represented as an edge weighted graph. Each vertex has a value indicating its score, each edge has a weight indicating the energy cost of traversing the edge. The agents cannot determine specific coordinates of the vertices, thereby making it hard to determine the direction an agent would go.

The distance between two connected vertices will be referred to as a *step*. The amount of steps away the agents can perceive, varies from 1 to 3. Whenever an agent perceives a vertex, he will remember the vertex, including the connecting vertices and the edges between them.

The vertices has to be probed to obtain information about how valuable the vertex is. Only the *Explorer* agent can probe vertices. If a vertex is not probed, the value of the vertex is set to be 1. The edges has to be surveyed to obtain information about how much it costs to traverse them. All agents can survey edges. The costs of traversing an edge is not depending on whether the edge is surveyed or not. Knowing the edge costs gives advantages when calculating paths for the agents to follow, i.e. shorter paths with lower costs.

2.2 Knowledge

Andreas

The agents share all their knowledge, that is, they have a centralized knowledge base. The agents therefore also have the same perception of the environment. The agents share all their new percepts before planning what they should do next. Having a centralized knowledge base, eliminates the need for communicating messages regarding perceptions of the environment, i.e. new vertices, new edges, opponent spotted.

3 Strategy

Morten

At game start the agents have no knowledge about the environment. They do not know where their fellow agents are located nor do they know where the opponent agents are. Therefore in order to be able to strategize properly for the game, the games is split into two phases. The two phases have been named the *Mapping* and *Zone Control Mode*.

June 2, 2014

Andreas Kjeldsen (s092638)
Morten Eskesen (s133304)
Artificial Intelligence and Multi-agent systems Peter Carlslund (s113998)

`Andreas`

3.1 Mapping

When the game starts, the first thing the agents focus on is mapping the environment. The Explorer agent focus on probing vertices, looking for unprobed vertices etc. The other agents walk around randomly looking for unsurveyed edges and opponents. The agents will only try to survey the surrounding edges if a specific amount of unsurveyed edges are visible. This avoids spending too much time surveying when only a few unsurveyed edges are near.

`Andreas &`
`Morten`

3.2 Zone Control Mode

At step 150 in the game, Zone Control mode is activated. This is based on the assumption that a great part of the environment is mapped at step 150. Zone Control mode is the phase where the agents defend a zone of the map, i.e. a subgraph of the entire graph. In Zone Control mode the agents only defend this zone and do not care about what the opponents are doing unless they are attacking us or threatening the zone in any way.

Zone Control mode is activated because the goal of the game is to achieve the highest score. The game grants points based on the zone score for the team. We have several algorithms that finds a high scoring zone in the graph, which we will cover in the section Zone Control. At each 150th step while in zone control mode, a zone recalculation is made, to see if a higher scoring zone can be found or if there is a way to expand the current zone.

3.2.1 Path Planning

During Zone Control mode the agents are to place themselves at specific vertices. Each agent must plan their way to their specific vertex. We have chosen to use Dijkstra's Algorithm for path planning. The path chosen is based on the cost of the edge and the amount of steps required to reach the specified vertex. We prefer paths with lower edge cost but also a path that requires fewer steps. A path could contain a small amount of edges to traverse but they could all be costly, therefore it might be advantageous to take a longer path with lower edge costs to avoid spending too much time recharging.

`Morten`

3.3 Reflection

Our strategy for the game is quite simple. When we enter zone control mode our agents are practically standing still with the hope that this would outscore our opponents. This might however not be the case for several reasons.

- If the agents haven't mapped, and probed, the highest scoring zone of the map and the opponent controls this zone. We have no plan in place to counter this scenario and will therefore be outscored by the opponent and loose.

- If the Explorer agents in the first phase haven't probed many nodes due to being disabled then we may end up defending a zone that does not achieve a high score.

June 2, 2014

Andreas Kjeldsen (s092638)
Morten Eskesen (s133304)
Artificial Intelligence and Multi-agent systems   Peter Carlslund (s113998)

This again is a scenario we have no plan in place to counter as the Explorer agents will not probe any nodes not in our controlled zone when entering zone control mode.

There are other scenarios we could consider but what one ultimately finds is that our strategy is quite naive. Also if the agents try to defend a zone where the opponents are, the agents will fight for this zone until a zone recalculation is made to find another zone. Our strategy would therefore be an obvious thing to improve if more time had allowed it, as we in general do not counter our opponents as much as we should.

4   Agents

Andreas

Each agent has a specific role. Depending on their assigned role, certain properties and abilities are available to the agent. The available agent roles are Explorer, Inspector, Repairer, Saboteur and Sentinel.

The agents make use of the *Belief-Desire-Intention* software model. This means that each agent has its own beliefs, its own desires and its own intentions. A belief is an assumption about the environment, though the agents are certain that a vertex will not change position, they are not certain that an opponent agent spotted will not move away. Therefore not all beliefs cannot be determined to be facts. A desire is something the agent desires to do. The role specific actions for the agents, determine their desires. The Explorer agent desires to probe unprobed vertices, the Saboteur agent desires to sabotage the opponent agent and so forth. The intentions of the agents are immediate intentions like, "I wish to go to vertex v233, on my I will have to pass through vertices v182 and v377".

An agent can be disabled, meaning it has 0 health left. A disabled agent is still able to `Goto` a vertex, `Recharge` to get energy and `Skip` its turn. A disabled Repairer agent can also use `Repair`. A disabled agent does not contribute to the zone scoring.

Each action available for the agents requires a certain amount of energy. The amount of energy required depends on the action. If an agent tries to perform an action that requires more energy than the agent has available, the action will fail. An agent can get more energy by using the `Recharge` action. If an agent is disabled it can still `Recharge` but the amount of energy recovered is less than if the agent wasn't disabled.

4.1   Agent Base

Andreas

The agents have some properties, functionality and actions in common. The agents start by interpreting their percepts, thereby gathering new knowledge and intel regarding the opponent team. The agents are able to determine if an agent is nearby based on the shared knowledge of the environment and the opponent positions. The agents are able to find a path from their current vertex to a goal vertex. The agents are able to make

distress calls, which indicates that they're in need of repairing.

In the event that an agent tries to perform an action but receives notification that they do not have enough energy to do so, the agent is forced to perform a `Recharge` before attempting to do the same action again. This avoids the case when an agent has an intention to do something, but not enough energy to do it, which could lead to an agent being stuck.

4.2 Explorer Agent

Andreas

The Explorer agent is the only agent capable of probing vertices. The Explorer agent prefers its possible actions in the following order: `Recharge` if energy is below $\frac{1}{3}$ of max, `Probe` if standing on unprobed vertex, `Goto` vertex found to be goal vertex, `Survey` edges if enough unsurveyed are found, `Goto & Probe` nearest unprobed vertex, `Recharge` if energy is not full otherwise `Skip`.

During the initial phase of the game, the agent probes as much as possible. During Zone Control mode, the agent will only probe unprobed vertices within the zone the agents are trying to control.

4.3 Inspector Agent

Andreas

The Inspector agent is the only agent capable of inspecting opponents agents. Inspecting an opponent agent reveals the agents role and properties, i.e. health and energy. The Inspector agent is able to perform the following actions: `Skip`, `Goto`, `Recharge`, `Survey` and `Inspect`.

Phase 1: Mapping

During the first phase the Inspector agent will help with the mapping by surveying unsurveyed edges. The Inspector has a limited visibility range of 1 and can therefore not see very far. Due to this, the Inspector can only `Survey` a few edges at a time. The visibility range of the Inspector also limits its ability to spot opponent agents. The Inspector agent will try to `Inspect` an opponent when its within the Inspector visibility range. An opponent will only be inspected once. If no unsurveyed edges have been found and no previously inspected opponent is nearby, the Inspector will `Goto` a random vertex. If the energy of the Inspector drops below 3, it will `Recharge`. If the health of the Inspector drops below 40% it will make a distress call.

Phase 2: Zone Control Mode

4.4 Repairer Agent

Andreas &
Peter

The job of the repairer agent, how we solve it and what it does during zone control mode etc.

Andreas &
Peter

4.5 Saboteur Agent

The job of the saboteur agent, how we solve it and what it does during zone control mode etc.

Morten

4.6 Sentinel Agent

The job of the Sentinel agent depends on phase of the game and whether or not the agent is disabled. The Sentinel agent is able to perform the following actions: `Skip`, `Goto`, `Parry`, `Recharge` and `Survey`.

Phase 1: Mapping

In the first phase the Sentinel will help with the mapping by surveying unsurveyed edges. The Sentinel has a visibility range of 3, meaning it can survey a lot of edges at once. If there is nothing to survey within its visibility range, the Sentinel will walk around randomly to further help the mapping process.

Being able to survive by itself is important for the Sentinel agent because we value it as one of the lesser valued agents. Therefore the Sentinel agent should burden the Repairer agents as little as possible. That is why the Sentinel agent will `Parry` or run away from an opponent if an opponent is close during the Mapping phase. The Sentinel agent will `Parry` a maximum of 5 times in a row, as we wanted to avoid the agent being stuck at some position. However there are things that rank higher in the hierarchy of assignments for the Sentinel. If it has less than $\frac{1}{4}$ of energy then it will always `Recharge` and if it receives a goal vertex it will `Goto` this vertex. The Sentinel will receive a goal vertex when Zone Control mode gets activated. If the Sentinel is disabled it will make a distress call.

Phase 2: Zone Control Mode

In the second phase the planning for the Sentinel agent becomes a lot more simple. The Sentinel will still always `Recharge` if it has less than $\frac{1}{4}$ energy left and `Goto` a goal vertex if it receives such a goal vertex. Otherwise the Sentinel will parry if an opponent is close to defend the zone and if no opponent is close it will `Recharge`. Note that in this phase there is no maximum amount of times the Sentinel can `Parry` it will `Parry` until its energy is less than $\frac{1}{4}$ of its total. As in phase 1 the Sentinel will make a distress call if it is disabled.

5 Planning

Andreas

Introduction to planning.

5.1 Planning Center

Andreas

We created a system, we'll be referring to as the *planning center*. The planning center keeps track of the actions the agents intent to perform. It refuses actions that would be a duplicate of an already planned action, while also preferring actions that are more beneficial than already planned actions. A more beneficial plan could be going to an unprobed vertex in as few steps as possible. The agents will only perform their action, when all agents have an action to perform that does not conflict with the others.

The agents plan their turn in a sequential order. The planning center therefore acts as a bidding system. The agents bid on the action they wish to perform, using how much the action would cost as their offer. Some actions are 'first come, first serve', these include probing of a specific vertex, surveying from a specific vertex, attacking a specific opponent agent or repairing a specific fellow agent. Other actions are given to the agent with the best offer. Best offer is determined based on the action. For going to and probing an unprobed vertex, the best offer is the lowest amount of steps required to reach the vertex. If an agent is outbid, the agent will have to come up with a new action to perform.

5.2 Distress Center

Andreas

Explain that agents can call for help, that repairers respond and that the distress center works with the planning center.

6 Zone Control

Morten

Zone control is where the agents defend a zone from their opponents. As mentioned earlier a good zone is a high scoring zone. In this section we will outline and compare the different algorithms we have implemented to find a high scoring zone, where only one is used in the final implementation. Finally we will explain how we guard the zone.

6.1 Algorithms

Andreas & Morten

There are a lot of approaches to achieving a good zone. One we have discussed a lot was finding a corner and then isolating this corner by making a "wall" in the map. This will cause a lot of nodes to be dominated by us and then it will probably achieve a high score even though our decision to isolate this corner had nothing to do with calculating the score of the nodes. However we have not found a way to identify corners which has lead us to consider other approaches. In the following sections we will describe these approaches and the algorithms we have implemented for this approach.

6.1.1 Isolated Subgraph

Morten

The isolated subgraph is an approach where we pick a high valued node and then we build a zone around this node. This node becomes the center of the zone. The algorithm

depends on how many agents we can use to build a zone. It will expand from the center node until we reach a number of nodes higher than the amount of agents. Then it will place the agents such that we dominate all the nodes inside this zone.

6.1.2 Max Sum Component

Andreas

Find connected components, take the one with maximum sum of specific size.

6.1.3 Simulated Annealing

Morten

Simulated annealing is an approach which locates a good approximation to the global optimum for a given function in a large search space. Simulated annealing is a good approach to locating a high scoring zone because the agents only have a certain amount of time to respond to the server and the map is a large search space. With other words the goal is to find an acceptably good solution in a fixed amount of time, rather than the best possible solution. The smart thing about simulated annealing is that it will probabilistically choose a solution if it is worse than the current solution found and always choose the new solution if it is better. The reason that it is smart to accept a worse solution is that it will never get stuck at a local optimum.

Our implementation of simulated annealing uses the previous algorithm described which finds a connect component of a fixed size and then expands this component for the purpose of getting a higher scoring zone. We discovered that the algorithm max sum component was really fast so instead of choosing a random node and try to expand around that we could just as well use this algorithm. The algorithm expands the zone by using the rules described for controlling a zone. It picks a random node which is not already dominated by us and is adjacent to one of the nodes we already have an agent placed at. Our implementation runs until the variable *temperature* > 1 where its initial value is 1000 and each iteration the temperature will "cool" by 0.025. The function to deem a solution acceptable (if the new solution is not a higher scoring zone) is Euler's number e raised to $(currentScore - newScore)/temperature$, where $currentScore$ it the score of the current created zone and $newScore$ is the score of the newly created zone.

$$e^{\frac{currentScore - newScore}{temperature}}$$

If this function is greater than a (pseudo) random number between 0 and 1 we accept the new solution.

6.1.4 Comparison

Morten

In this section we will compare the algorithm by simulation a game versus only one opponent which will only skip so we can see the outcome of the algorithms in the most optimal setting.

Algorithm	Score
Isolated Subgraph	0
Max Sum Component	0
Simulated Annealing	0

6.2 Guarding The Zone

Morten

We guard our zone by having all the agents which are able to parry parry such that they survive when enemies attack them. If opponents are able to intrude our zone we attack them with the hope of killing them and regaining control of our zone. We also guard our zone by repairing all the agents that become disabled as the result of an intrusion or attack on our zone in order not to lose points.

7 Conclusion

Andreas &
Morten &
Peter

Conclude on the problems, state that we solved most of them if not all.

7.1 Flaws

Peter

A shit load of flaws in our solution and hopefully how to remedy them had we had more time!

References

[1] Stuart J. Russel, Peter Norvig: *Artificial Intelligence: A Modern Approach*. 3rd Edition, Pearson, 2010.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: *Introduction To Algorithms*. 3rd Edition, MIT Press, 2009.

[3] Robert Sedgewick, Kevin Wayne: *Algorithms*. 4th Edition, Addison Wesley, 2011.