(Adversarial) Multi-Agent Path Finding

Marika Ivanová

Department of Theoretical Computer Science and Mathematical Logic

Charles University in Prague
Faculty of Mathematics and Physics

ivanova@ktiml.mff.cuni.cz

October 22, 2020

matfyz

2020-10-22

Inf UFG

Hello and welcome to open day at Faculty of Mathematics and Physics. In this video we are going to talk about Adversarial Multi-Agent Path Finding, which is a topic in the area of Artificial Intelligence. First we will talk about traditional Multi-Agent Path Finding and then we show how to extend this problem by an adversarial element.

Multi-Agent Path Finding

Problem description

- A group of agents (robots) deployed in a given environment
- Each agent has a defined initial and target location
- **Task:** relocate the agents to their targets while avoiding collisions

Objective function:

- Minimum makespan
- Minimum total distance
- Minimum total arrival time

Multi-Agent Path Finding
Problem description
- A group of agents (robots) deployed in a given environment
- Each agent has a defined initial and target location
- Task: relocate the agents to their targets while avoiding collisions

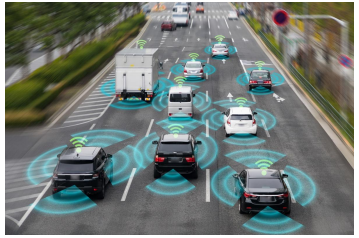
Objective function:
- Minimum makespan
- Minimum total distance
- Minimum total arrival time

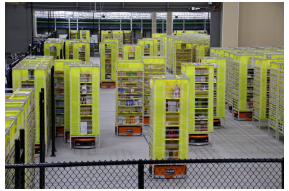
Let us consider a group of agents, for example robots deployed in a given environment with obstacles. Each agent has a defined initial and target location. Our task is to navigate the agents to their respective target positions, so that they do not collide with each other or with the obstacles.

We would also like to do it as best as possible. There are several criteria for assessing quality of a solution. The most common is to use so called minimum makespan, in which minimizes the time of the last arriving agent. There is also a possibility to minimize the total number of moves, which means that the agents should cover the shortest possible distance. Another option is to minimize total time that agents spend on their way.

Motivation

- Moving goods in warehouses
- Manipulation of shipping containers
- Navigation of a group of autonomous vehicles



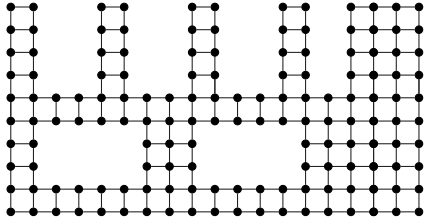


Motivation
- Moving goods in warehouses
- Manipulation of shipping containers
- Navigation of a group of autonomous vehicles

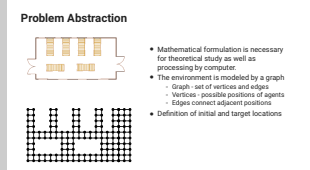
This problem is motivated by various practical applications. Recently, robots have been widely used in large warehouses, where they move goods or racks from one place to another. Another logistic application is moving shipping containers. In this case, agents are the containers. It is clear that moving such large objects should be done as efficient as possible. Another hot topic is autonomous vehicle control, particularly in traffic jams.

Here we have a short video showing how Kiwa robots move in Amazon warehouse. We can see a group of robots that need to relocate to a certain place and must not collide.

Problem Abstraction



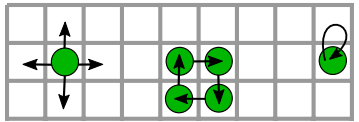
- Mathematical formulation is necessary for theoretical study as well as processing by computer.
- The environment is modeled by a graph
  - Graph - set of vertices and edges
  - Vertices - possible positions of agents
  - Edges connect adjacent positions
- Definition of initial and target locations

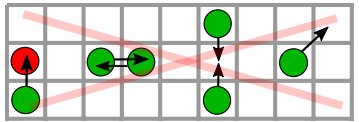


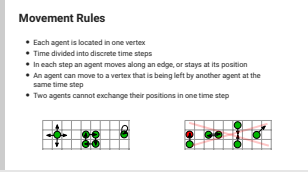
In order to study the problem either theoretically or conduct simulations, it is necessary to define it formally. The environment in which agents are deployed is modeled by a graph. A graph is a mathematical structure consisting of a set of Vertices and a set of Edges. Vertices are points and edges are links between them. The vertices represent potential locations of agents, and the edges connect adjacent locations. Here in the figure above we can see a plan of some warehouse. We have one vertex for each positions in which a robot can stand We assume that the robots can move in 4 directions, and so we obtain a grid graph like in the figure below.

Movement Rules

- Each agent is located in one vertex
- Time divided into discrete time steps
- In each step an agent moves along an edge, or stays at its position
- An agent can move to a vertex that is being left by another agent at the same time step
- Two agents cannot exchange their positions in one time step

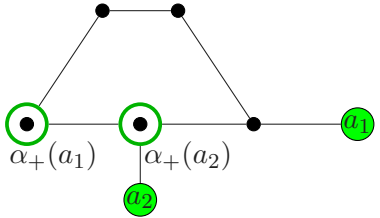






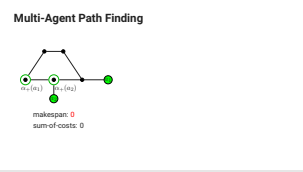
Next, it is necessary to define rules for agents' movement. Each agent is placed in exactly one vertex. Continuous time is divided into discrete time steps, for example seconds. In each time step an agent can move to an adjacent node along an edge, or stay at its position. An agent can also move to a vertex that is being left by another agent at the same time step. An exception to this rule is the case when two agents attempt to exchange their positions - that is not allowed.

Multi-Agent Path Finding



makespan: 0

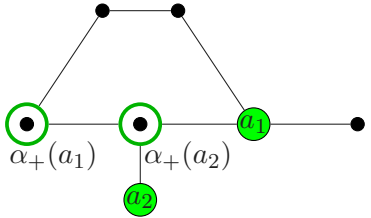
sum-of-costs: 0



Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

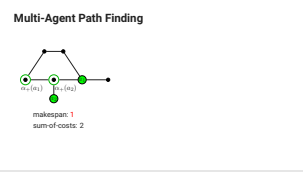
Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

Multi-Agent Path Finding



makespan: 1

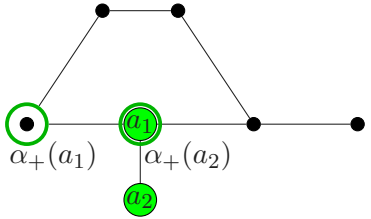
sum-of-costs: 2



Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

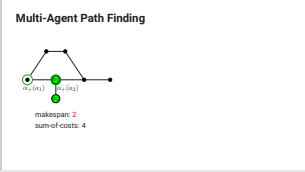
Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

Multi-Agent Path Finding



makespan: 2

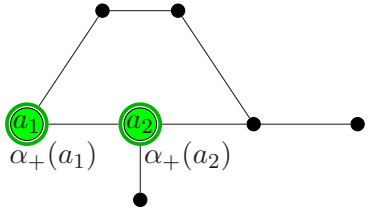
sum-of-costs: 4



Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

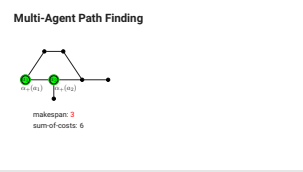
Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

Multi-Agent Path Finding



makespan: 3

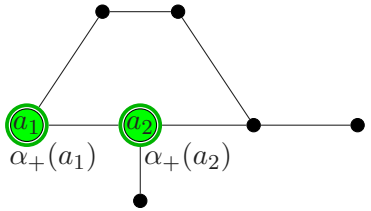
sum-of-costs: 6



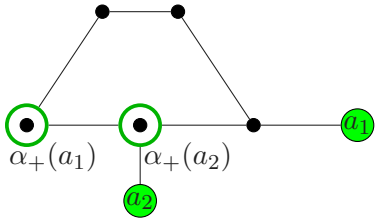
Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

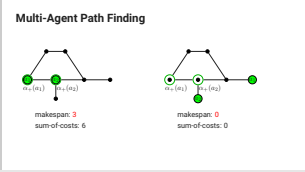
Multi-Agent Path Finding



makespan: 3
sum-of-costs: 6



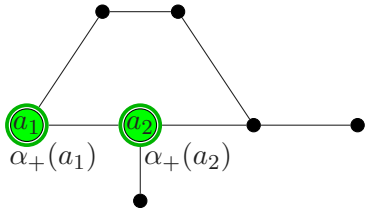
makespan: 0
sum-of-costs: 0



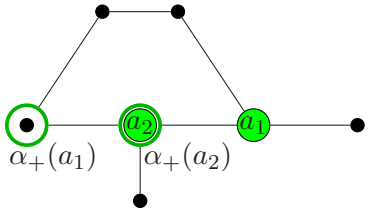
Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

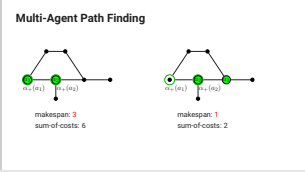
Multi-Agent Path Finding



makespan: 3
sum-of-costs: 6



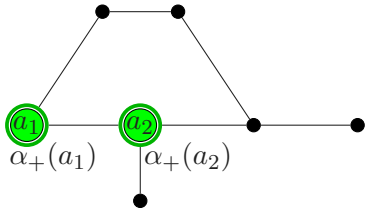
makespan: 1
sum-of-costs: 2



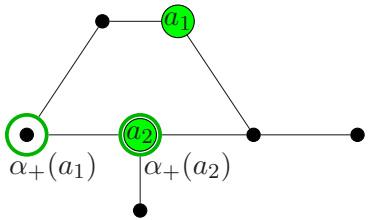
Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

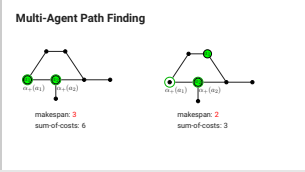
Multi-Agent Path Finding



makespan: 3
sum-of-costs: 6



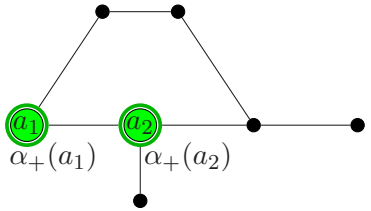
makespan: 2
sum-of-costs: 3



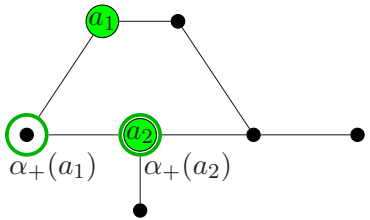
Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

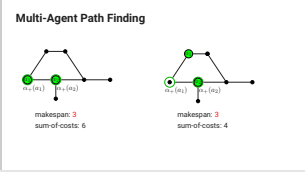
Multi-Agent Path Finding



makespan: 3
sum-of-costs: 6



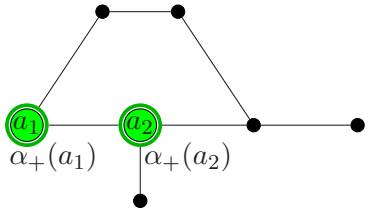
makespan: 3
sum-of-costs: 4



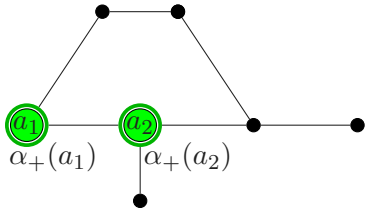
Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

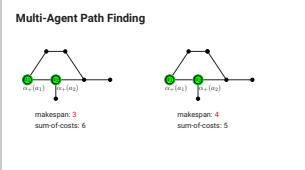
Multi-Agent Path Finding



makespan: 3
sum-of-costs: 6



makespan: 4
sum-of-costs: 5



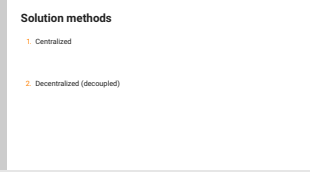
Now we will see an example of a simple instance with two agents. The Vertices in green circles are targets of the agents. We will see what is the optimal solution for the shortest arrival time of the last agent, called minimum makespan. Although a2 stands right next to its target, it will not move there in the first step. Instead of that, A1 rushes via the shortest path towards its target. A2 thus gives way to A1 When A1 is performing the last move A2 can simultaneously move to its target. This solution has makespan 3, while total arrival time of all agents is 6.

Now we will see a solution with a longer makespan, but shorter total arrival time. In this solution A2 immediately moves to its target, and A1 has to go along the longer upper path. This solution has total arrival time 5, which is optimal. On the other hand, makespan is 4, which can be done better as we have seen in the previous example.

Solution methods

1. Centralized

2. Decentralized (decoupled)

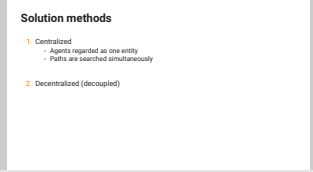


Solution methods are divided into two main groups - centralized and decentralized. In centralized approaches, whole group of agents is considered as a single entity. Paths are then searched simultaneously for all agents. In centralized methods are paths searched individually for each agent. Centralized methods typically yield a better solution than decentralized methods, however, centralize approaches are often computationally more challenging.

Solution methods

1. Centralized
   - Agents regarded as one entity
   - Paths are searched simultaneously

2. Decentralized (decoupled)

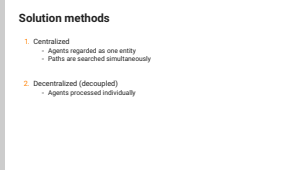


Solution methods are divided into two main groups - centralized and decentralized. In centralized approaches, whole group of agents is considered as a single entity. Paths are then searched simultaneously for all agents. In centralized methods are paths searched individually for each agent. Centralized methods typically yield a better solution than decentralized methods, however, centralize approaches are often computationally more challenging.

Solution methods

1. Centralized
   - Agents regarded as one entity
   - Paths are searched simultaneously

2. Decentralized (decoupled)
   - Agents processed individually



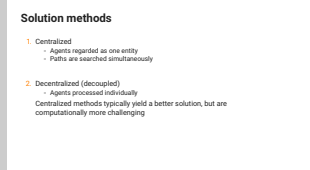
Solution methods are divided into two main groups - centralized and decentralized. In centralized approaches, whole group of agents is considered as a single entity. Paths are then searched simultaneously for all agents. In centralized methods are paths searched individually for each agent. Centralized methods typically yield a better solution than decentralized methods, however, centralize approaches are often computationally more challenging.

Solution methods

1. Centralized
   - Agents regarded as one entity
   - Paths are searched simultaneously

2. Decentralized (decoupled)
   - Agents processed individually

   Centralized methods typically yield a better solution, but are computationally more challenging



Solution methods are divided into two main groups - centralized and decentralized. In centralized approaches, whole group of agents is considered as a single entity. Paths are then searched simultaneously for all agents. In centralized methods are paths searched individually for each agent. Centralized methods typically yield a better solution than decentralized methods, however, centralize approaches are often computationally more challenging.

Adversarial Multi-Agent Path Finding (AMAPF)

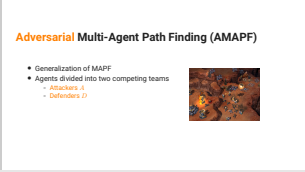
- Generalization of MAPF
- Agents divided into two competing teams

Adversarial Multi-Agent Path Finding (AMAPF)
• Generalization of MAPF
• Agents divided into two competing teams

A generalization of the problem introduces an adversarial element in multi agent path finding. Agents are divided into two competing teams called Attackers and defenders. Attackers' task is the same as we have seen so far: they need to navigate agents to their target positions while avoiding collisions. Defenders on the other hand try to protect these targets from the attackers. The problem becomes a two player games where teams as players take turns.

Adversarial Multi-Agent Path Finding (AMAPF)

- Generalization of MAPF
- Agents divided into two competing teams
  - Attackers A
  - Defenders D



A generalization of the problem introduces an adversarial element in multi agent path finding. Agents are divided into two competing teams called Attackers and defenders. Attackers' task is the same as we have seen so far: they need to navigate agents to their target positions while avoiding collisions. Defenders on the other hand try to protect these targets from the attackers. The problem becomes a two player games where teams as players take turns.

Adversarial Multi-Agent Path Finding (AMAPF)

- Generalization of MAPF
- Agents divided into two competing teams
  - Attackers A
  - Defenders D
- Attackers aim to reach their target positions
- Defenders protect the targets from attackers

Adversarial Multi-Agent Path Finding (AMAPF)
- Generalization of MAPF
- Agents divided into two competing teams
  - Attackers A
  - Defenders D
- Attackers aim to reach their target positions
- Defenders protect the targets from attackers

A generalization of the problem introduces an adversarial element in multi agent path finding. Agents are divided into two competing teams called Attackers and defenders. Attackers' task is the same as we have seen so far: they need to navigate agents to their target positions while avoiding collisions. Defenders on the other hand try to protect these targets from the attackers. The problem becomes a two player games where teams as players take turns.

Adversarial Multi-Agent Path Finding (AMAPF)

- Generalization of MAPF
- Agents divided into two competing teams
  - Attackers A
  - Defenders D
- Attackers aim to reach their target positions
- Defenders protect the targets from attackers
- Two player game
- Teams take turns in each step

A generalization of the problem introduces an adversarial element in multi agent path finding. Agents are divided into two competing teams called Attackers and defenders. Attackers' task is the same as we have seen so far: they need to navigate agents to their target positions while avoiding collisions. Defenders on the other hand try to protect these targets from the attackers. The problem becomes a two player games where teams as players take turns.

Adversarial Multi-Agent Path Finding (AMAPF)

- Generalization of MAPF
- Agents divided into two competing teams
  - Attackers A
  - Defenders D
- Attackers aim to reach their target positions
- Defenders protect the targets from attackers
- Two player game
- Teams take turns in each step
- **Motivation:** military and security simulations, video game industry

Adversarial Multi-Agent Path Finding (AMAPF)
- Generalization of MAPF
- Agents divided into two competing teams
  - Attackers A
  - Defenders D
- Attackers aim to reach their target positions
- Defenders protect the targets from attackers
- Two player game
- Teams take turns in each step
- Motivation: military and security simulations, video game industry

A generalization of the problem introduces an adversarial element in multi agent path finding. Agents are divided into two competing teams called Attackers and defenders. Attackers' task is the same as we have seen so far: they need to navigate agents to their target positions while avoiding collisions. Defenders on the other hand try to protect these targets from the attackers. The problem becomes a two player games where teams as players take turns.

Adversarial Multi-Agent Path Finding (AMAPF)

Victory of attackers: all agents from A reach their targets
Problem: find a winning strategy for A

Adversarial Multi-Agent Path Finding (AMAPF)
Victory of attackers: all agents from A reach their targets
Problem: find a winning strategy for A

The team of attackers wins in case all its agents manage to arrive at their targets. Our task is to find a winning strategy for the team of attackers. A winning strategy means that in each step when attacker is to move, they play such a move that defenders cannot prevent victory of attackers no matter how they play. Theoretical study of this problem shows that already the decision whether a winning strategy for attackers exists is very hard, specifically PSPACE-hard. Recently, it seems however that it is even harder, which we call EXPTIME hard You will find out more about these strange terms when you come to study at our faculty.

Adversarial Multi-Agent Path Finding (AMAPF)

Victory of attackers: all agents from A reach their targets

Problem: find a winning strategy for A

Winning strategy

In each step which is A's turn, A plays such a move that leads to its victory regardless of the response of defenders D.

The team of attackers wins in case all its agents manage to arrive at their targets. Our task is to find a winning strategy for the team of attackers. A winning strategy means that in each step when attacker is to move, they play such a move that defenders cannot prevent victory of attackers no matter how they play. Theoretical study of this problem shows that already the decision whether a winning strategy for attackers exists is very hard, specifically PSPACE-hard. Recently, it seems however that it is even harder, which we call EXPTIME hard You will find out more about these strange terms when you come to study at our faculty.

Adversarial Multi-Agent Path Finding (AMAPF)

Victory of attackers: all agents from A reach their targets

Problem: find a winning strategy for A

Winning strategy
In each step which is A's turn, A plays such a move that leads to its victory regardless of the response of defenders D.

- The decision whether there exists a winning strategy is very hard (PSPACE-hard)

The team of attackers wins in case all its agents manage to arrive at their targets. Our task is to find a winning strategy for the team of attackers. A winning strategy means that in each step when attacker is to move, they play such a move that defenders cannot prevent victory of attackers no matter how they play. Theoretical study of this problem shows that already the decision whether a winning strategy for attackers exists is very hard, specifically PSPACE-hard. Recently, it seems however that it is even harder, which we call EXPTIME hard You will find out more about these strange terms when you come to study at our faculty.

Adversarial Multi-Agent Path Finding (AMAPF)

Victory of attackers: all agents from A reach their targets

Problem: find a winning strategy for A

Winning strategy
In each step which is A's turn, A plays such a move that leads to its victory regardless of the response of defenders D.

- The decision whether there exists a winning strategy is very hard (PSPACE-hard)
- It is probably even harder (EXPTIME-hard)

The team of attackers wins in case all its agents manage to arrive at their targets. Our task is to find a winning strategy for the team of attackers. A winning strategy means that in each step when attacker is to move, they play such a move that defenders cannot prevent victory of attackers no matter how they play. Theoretical study of this problem shows that already the decision whether a winning strategy for attackers exists is very hard, specifically PSPACE-hard. Recently, it seems however that it is even harder, which we call EXPTIME hard You will find out more about these strange terms when you come to study at our faculty.

Adversarial Multi-Agent Path Finding (AMAPF)

Victory of attackers: all agents from A reach their targets

Problem: find a winning strategy for A

Winning strategy
In each step which is A's turn, A plays such a move that leads to its victory regardless of the response of defenders D.

- The decision whether there exists a winning strategy is very hard (PSPACE-hard)
- It is probably even harder (EXPTIME-hard)
- You will get to know what that means when you join us :)

The team of attackers wins in case all its agents manage to arrive at their targets. Our task is to find a winning strategy for the team of attackers. A winning strategy means that in each step when attacker is to move, they play such a move that defenders cannot prevent victory of attackers no matter how they play. Theoretical study of this problem shows that already the decision whether a winning strategy for attackers exists is very hard, specifically PSPACE-hard. Recently, it seems however that it is even harder, which we call EXPTIME hard You will find out more about these strange terms when you come to study at our faculty.

Demonstration of agents’ movement in an AMAPF instance

Demonstration of agents’ movement in an AMAPF instance

Now we will have a look at a short visualisation of movement of agents in Adversarial multi agent path finding. Three light green attackers 4 5 and 6 at the top has three dark green targets at the bottom. Agent 3 has its target in the remaining dark green node. Let us have a look at what movement the green attackers have to perform, so that the red defenders would not prevent them from reaching the targets.

Solution methods

- In a typical instance, not all attackers manage to reach their target
- We therefore focus on occupying as many targets as possible

Greedy

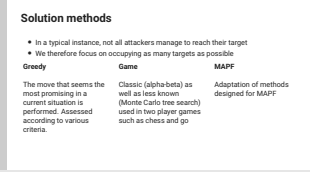
The move that seems the most promising in a current situation is performed. Assessed according to various criteria.

Game

Classic (alpha-beta) as well as less known (Monte Carlo tree search) used in two player games such as chess and go

MAPF

Adaptation of methods designed for MAPF



In the majority of instances it is not possible that all attackers reach their targets. We therefore try to maximize the number of targets captured by attackers. We propose algorithms that get some instance as an input, and whenever it is attacker's turn, it selects its move. So far we have evaluated three types of algorithms The simplest greedy methods perform the move that seems to lead tho the best position. This can often be a bit myopic. On the other hand, game algorithms such as minimax alpha beta and others try to look a bit further ahead, and perform such a move that leads to the best possible result in a certain time frame. There are many approaches designed For traditional multi agent path finding It is also possible to adapt them so that they contain the adversarial element.

Future research

MAPF has been studied extensively since last several years

That is not the case for AMAPF

Possible directions of future research:

- Computational complexity of special cases
- More detailed evaluation of solution techniques
- Design of new solution methods
- Strategy for the team of defenders
- Study of modified problems (e.g., not all agents have targets)

Future research

MAPF has been studied extensively since last several years
That is not the case for AMAPF

Possible directions of future research:

- Computational complexity of special cases
- More detailed evaluation of solution techniques
- Design of new solution methods
- Strategy for the team of defenders
- Study of modified problems (e.g., not all agents have targets)

Adversarial Multi agent path finding is a topic with a good research potential It can be studied both theoretically and practically, and interesting results are awaited in both directions. The results are expected to be published on international conferences on artificial intelligence and related topics.

Sources

1. Adversarial Cooperative Path-Finding: Complexity and Algorithms, 2014 IEEE 26th International Conference on Tools with Artificial Intelligence
2. Adversarial Cooperative Path-Finding: A First View, AAAI (Late-Breaking Developments) 2013
3. www.koupy.net/graphrec.php
4. www.therobotreport.com
5. www.smartcitiesworld.net

Sources

1. Adversarial Cooperative Path-Finding: Complexity and Algorithms, 2014 IEEE 26th International Conference on Tools with Artificial Intelligence
2. Adversarial Cooperative Path-Finding: A First View, AAAI (Late-Breaking Developments) 2013
3. www.koupy.net/graphrec.php
4. www.therobotreport.com
5. www.smartcitiesworld.net

So this was a short presentation about adversarial multi agent path finding, thank you very much for watching.