Learning in Multi-Agent Path Finding

Written by Bc. Petra Vysušilová
vysusilova.petra.m@gmail.com

Abstract

This paper offers an overview of current work in the applications of various machine learning techniques on the Multi-Agent Path Finding (MAPF) problem. It also offers improvement suggestions and possible paths for further research.

Introduction

Multi-Agent Path Finding (MAPF) recently appeared to be an interesting problem with many real-world applications. It can serve not only for industrial use in automated warehouses or autonomous cars but also as a big help for emergencies in environments that are toxic, dangerous, or difficult to access for humans (Sartoretti et al. 2019a) (Stern et al. 2019).

Classical Multi-Agent Path Finding task and many of its variants are analyzed and defined in (Stern et al. 2019). As it is stated there, MAPF problem is not clearly defined – in different papers, various metrics are optimized and different conflict types are taken into account. In this work, included research papers are not focusing on one standardized problem type, as they assume various constraints and have diverse motivations in mind. For these purposes, I decided to treat MAPF as the task of finding collision-free paths for all agents (virtual or embodied) from their starting point to their respective goals. Sometimes it is enough to find a valid solution, other times we also want to optimize time or path length.

The solution for MAPF-related problems is sought via many learning methods, especially various types of reinforcement learning (RL), neural networks (NN), and genetic algorithms (GA). The major approach is, however, centralized planning and its working solutions are already deployed to real industrial use. What is the motivation behind exploring machine learning methods in this context? Even finding an approximate solution to MAPF belongs to NP-hard problems. As number of agents increases, classical planning runs into a problem with the curse of dimensionality, because the size of joint action-state space is exploding. Machine learning methods, especially RL and NN, are hot topics and are expected to be more capable of scalability. Another reason for learning is the fact that centralized planning solutions do not take into account some technical inaccuracies in robotic movement (not perfectly constant speed) or computational resources do not suffice for a more complex model of the world. Such model can be a graph containing few places and edges weighted by distances between them, while real robots need some extra time for turning in corners, velocity is different on crooked paths etc.

Reinforcement learning is a set of techniques for finding the most rewarded behavior. Although there are modifications for continuous time, standard RL is operating in discrete time steps. Every time step, the agent can observe the environment state, choose one action, and obtain a reward R (which can be zero). This process can be described as a partially observable Markov decision process (POMDP), which brings the important assumption that state in time t does not depend on any other previous steps, given a state and an action in time $t-1$ (Sutton and Barto 2018).

POMDP is a quadruple (S, A, p, γ):
S ... set of states
A ... set of actions
$\gamma \in (0, 1 > \ldots$ discount factor
$p_i(S_{t+1} = s^i, R_{t+1} = r | S_t = s, A_t = a)$ ... probability of following state will be s^i and reward will be equal to r given the current state s and an action a.
For every time step, $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$ is discounted sum of rewards and the goal of learning is to learn policy

$$\pi : state \rightarrow action$$

which maximizes $\mathbf{E}(R_0)$.

Problem specification

Before a presentation of chosen interesting recent research results, a brief overview of different approaches will be presented in this section, but one terminology note at first. Because methods described bellow use different mechanisms for making decisions about the next step(s), the term *decision-maker* will be used for all algorithms in the following text. In some works, the term *planner* can be encountered, but the term is not used in this way as methods do not fit the definition of a planner in the classical-planning-way sense.

Problem space definition

The main factor influencing the complexity of the problem is what the decision-maker knows about the space. In the *coupled* (centralized) approach, all the information available to some robots is considered together at the moment of computation. This does not mean that the whole plan must be created in advance. The planning can happen at each step. Further, this does not imply, that the decision-maker actually needs to know the whole truth about the reality as the decisions can be based on estimates. This approach is computationally demanding. On the contrary, in the *decoupled* approach, the path is created for each agent separately. This way has an obvious problem with conflicts because there is simply no guarantee that the shortest paths for two robots will not collide. For this reason, *dynamically coupled* decision-makers seem to be a good solution. They are operating in a decoupled space until they need to solve a conflict. Then they expand the space necessarily.

Available information

The information available at the moment of decision can differ. Optimal results are obtained if a **complete information about the world** is presented. Unfortunately, not just that scalability problem appears, but it is also in some cases unrealistic to have all information about the world.

Focus of view consists of information from all its sensors filtered based on its relevancy or distance from an agent. In the contrast to the use of information about the whole world, we can decide to use just the information collected from the FOV of agents.

Again, there exists a compromise solution, where only some **information is shared**, so the space of possible solutions is not so big, but agents have a wider knowledge of the environment.

Possible combinations

All types of information can be then used in a coupled or decoupled approach. Most of the later discussed papers focus on the decoupled approach together with incomplete information. In this case, the goal of learning could be to teach an agent to move in space effectively, to solve conflicts, and to not block the path between the other agent and its goal. This method of course cannot be significantly better than the behavior of humans. There is no guarantee of the optimality with respect to time or sum of path lengths, but it is a picture of many real applications (motion in unknown space or with other agents whose intentions are not known) and it demands fewer resources.

Current Work

Now is the time for presenting recent research focusing on various aspects of the problem and various solution techniques. Recent research can be divided into following groups – reinforcement learning policy, communication, nature-inspired algorithms, and agent's motion, which will be described in this section.

Reinforcement learning policy

In this section, papers that develop reinforcement learning policies are described. As was mentioned above, RL focuses on finding policies for the selection of the best next step.

PRIMAL The key algorithm in this section is *Pathfinding via Reinforcement and Imitation Multi-Agent Learning* (PRIMAL) (Sartoretti et al. 2019a). It is a planning algorithm based on actor-critic reinforcement learning (Barto, Sutton, and Anderson 1983) and imitation learning. A detailed description will be offered, as it is a core paper for MAPF learning.

Motivation Centralized planning is able to find optimal paths for all agents, but it can take a lot of time for big environments with many robots. During the execution can happen some unpredictable things and the planner is forced to replan new paths, which again takes a lot of time and can make this algorithm to be inapplicable in practice. The goal of this paper is to find an algorithm that can quickly decide about the next action in every step based on actually available information. For this purpose, the considered world is only partially observable and each agent is behaving according to its policy in every step (decoupled approach).

World definition and available information Agents are trained in a discrete partially observable grid world. Each agent knows its FOV (in the paper, 10x10 squares was used), a direction towards its goal, and an euclidean distance to the goal (see Figure 1).

Technologies The algorithm is based on deep RL, specifically on Asynchronous Advantage Actor-Critic (A3C) algorithm (Mnih et al. 2016). A3C is a reinforcement learning algorithm based on deep neural networks. As can be seen in Figure 2, it consists of many parallel agents, each with its own copy of an environment. Thus, every agent can learn independently on other agents, which is more robust and effective. It is an actor-critic method, so this network has two goals – estimate a value function for every state and create a good policy for every state, i.e. estimate true values for every state-action pair and select the best action.

Learning itself is performed via distributed reinforcement learning (Sartoretti et al. 2019b). Each agent has its own copy of a neural network, whose weights are synchronized with one common network after each step. Gradients obtained from different agents are pushed to the common network, so all agents are learning together.

The architecture of the neural network can be seen in figure 3. It is composed of convolution layers processing FOV and fully connected layers for goal direction. Preprocess features are then input into LSTM unit. The inputs of the network are all information agent has (4x10x10 tensor with FOV layers) and information about its goal. In contrast to A3C, where the network learns only policy and value functions, this network has also a *blocking* output – it is trained to indicate whether the position of the agent is blocking another agent on its way to the goal. The agent's position is considered to be blocking if another agent cannot reach its goal or is delayed by 10 or more steps. This potential delay is calculated just from a path found by A* (Hart, Nilsson, and Raphael 1968) for one agent, so it is a heuristic, not a real de-

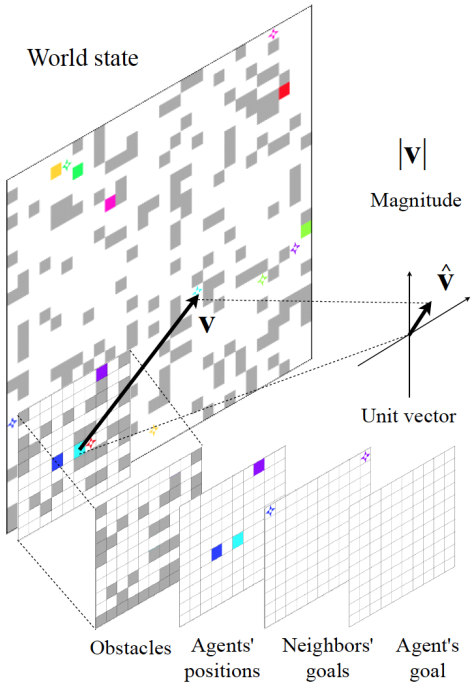

Figure 1: All available information for an agent. Agent is on light blue position in the middle of the grid. Agent's goal position is explicitly known only if it is in agent's focus of view, but agent always has information about the direction and a distance to its goal. *source: Fig.6 of (Sartoretti et al. 2019a)*

lay caused by learned policy (obviously, if it would be possible to compute shortest paths for every agent exactly and quickly, no RL is needed). The agent itself does not know the goals of other agents, but blocking output causes the penalty, which affects the loss function and then learned policy. This penalty is the part of the solution for the most difficult thing – move away from their own goal (when the agent obtains maximal reward) and does not block another agent on its path to the goal. Two other things are contributing to the achievement of such behavior, imitation learning of expert planner (ODrM* (Wagner and Choset 2015)) and very dense training data. Expert planner shows paths worth to imitate and dense training data cause more conflicts and therefore more learning opportunities. A right setting of reward function can by often tricky in RL. In this case, the authors selected the reward function structure described in table 1. Random environment size and obstacles density sampling at the beginning of each episode exposes agents to many difficult situations and helps to learn more generalized policy.

Training At the beginning of every episode, a randomly sized world is generated. Smaller worlds are more probable because authors want agents to be exposed to the most difficult tasks. An obstacle density is also generated in random and obstacles and agents are placed uniformly. There is just checked, that none of the goals is separated from its agent by obstacles (unreachable).

Results Results are compared to three planners - ORCA

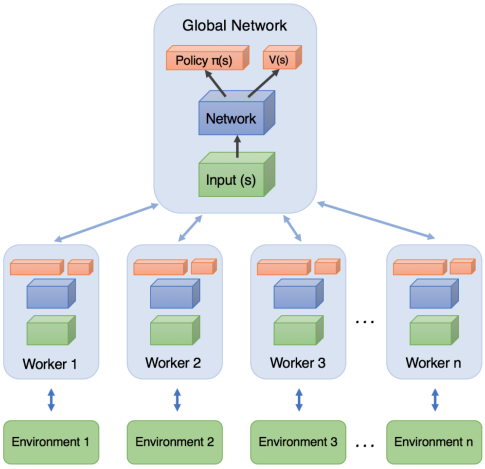

Figure 2: An overall architecture of A3C. Every agent interacts with its own separate copy of an environment. All agents contribute to one common network, which learns value and policy functions. *source: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2*

Table 1: PRIMAL structure of the reward function.

Action	**Reward**
Move	-0.3
Agent collision	-2.0
No movement (on/off goal)	$0.0/-0.5$
Finish episode	$+20$
Blocking penalty	-0.2

(Van Den Berg et al. 2011), ODrM* (Wagner and Choset 2015) and CBS (Sharon et al. 2015). Figure 4 shows that all three main algorithm features (imitation learning, reward structure, and density of obstacles) together are important for reasonable performance. ODrM* needed minutes for replanning, while PRIMAL can decide the next action in less than 0.2s.

As shown in figure 5, PRIMAL is quite good for environments with low obstacle density and very good for obstacle-free worlds. It is outperformed in worlds with a high density of obstacles. This problem was addressed in future work (described later in this text).

GLAS One of the most recent papers is GLAS (Rivière et al. 2020) – *Global-to-Local Safe Autonomy Synthesis for Multi-Robot Motion Planning with End-to-End Learning*.

Motivation This paper combines both centralized and decoupled approach with their advantages. As in the case of PRIMAL, GLAS tries to reduce the time necessary to recalculate new paths when something unpredicted happens. For this reason, the output is a policy for every step action. It also incorporates some part (called *safety modul*) which ensures

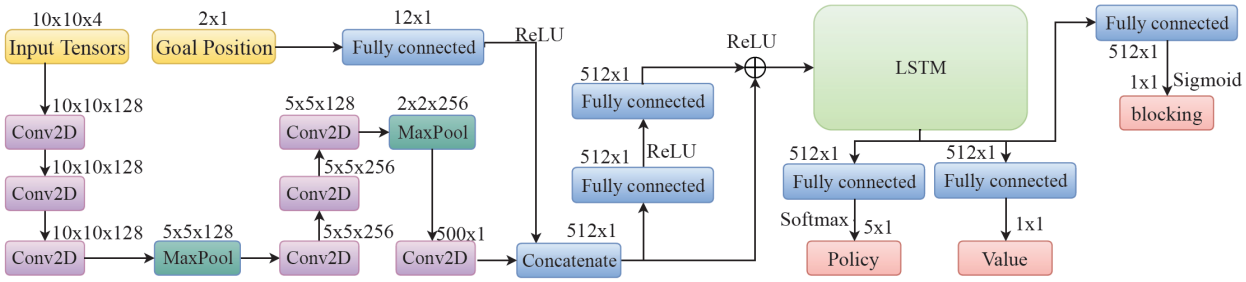

Figure 3: PRIMAL neural network architecture. Inputs are composed by agent's observation and a goal position. Outputs is policy for actions, value (serves as critic in the A3C) and an indicator of blocking positions.
source: Fig.3 of (Sartoretti et al. 2019a)

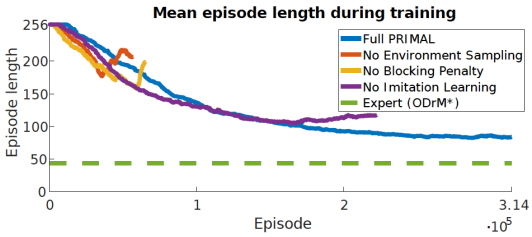

Figure 4: Primal episode length comparison.
Mean episode length during training, lower is better. The dotted line shows the baseline, obtained from the expert ODrM* planner. When we remove either environment sampling, the blocking penalties, or imitation learning from our approach, the policy converges to a worse solution. *source: Fig.5 of (Sartoretti et al. 2019a)*

that potential collision will be solved and such a guarantee is what a centralized approach is valued for.

World definition and available information GLAS operates in continuous state/action space. The state is represented by position or position and velocity and possible actions are velocity or an acceleration. It could be categorized as semi-decoupled approach with incomplete information.

Technologies Using continuous state/action space is allowed by using Deep Sets (Zaheer et al. 2017), which are also more efficient than convolution networks used by PRIMAL.

The safety module is fully differentiable, so it can become a part of the loss function and help in policy learning.

Training Final loss function for a network is a combination of RL loss function and safety module, thus all derivations in backpropagation are computed with respect to this new compound function. It was found that is better to use this composition than to train network concerning RL loss function only. This method causes nice cooperation with the safety module, although the safety module has no trainable parameters and therefore is not updated by backpropagation. The safety module guarantees no collisions between agents. For information about training data generation, see Figure 7.

Similarly to PRIMAL, GLAS also uses imitation learning, although the application is different. It tries to transfer good global planner experience to local. To achieve this, at the input of the network, all global-only information from each expert trajectory is masked out, so the policy is learned only on the data available for an agent.

Results Performance experiments are performed on a relatively small count of robots (with a maximum of 16 robots) in comparison to the high scalability of PRIMAL. Results, however, turned out to be better than actual state-of-the-art ORCA (see Fig. 8) and as it is able to efficiently solve the problem in continuous space with a safety guarantee, it is definitely worth mentioning.

(Cao, Sun, and Yan 2019) One of the possible real uses of RL methods is autonomous underwater vehicles (AUV). On such problem focuses following paper (Cao, Sun, and Yan 2019).

Motivation The motivation behind this paper includes military purposes (underwater detection of enemy infiltrators, their location and capture), rescue operations, scientific exploration, or searching for resources. The problem does not fit MAPF definition, it belongs to target searching (TS) as the goal of these vehicles is to search the water for an unknown moving target(s). It means that the goal position is not defined in advance, the position is the part of the solution result. Even so, the path to the goal can be long and ineffective, obstacles and collisions can occur, and plan for the path should dynamically changes, which is the reason for including the paper in this overview.

World definition and available information The algorithm operates on the grid map of size 64x64, where every tile can be in one of three states - open, occupied, or unknown, other inputs are the location of all possible boundaries and the position of AUV.

Technologies This algorithm works in three stages:

- create environment grid map from sonar information,
- search module is responsible for finding a potentially interesting location,
- navigation module than calculates the path from a current position to the selected goal.

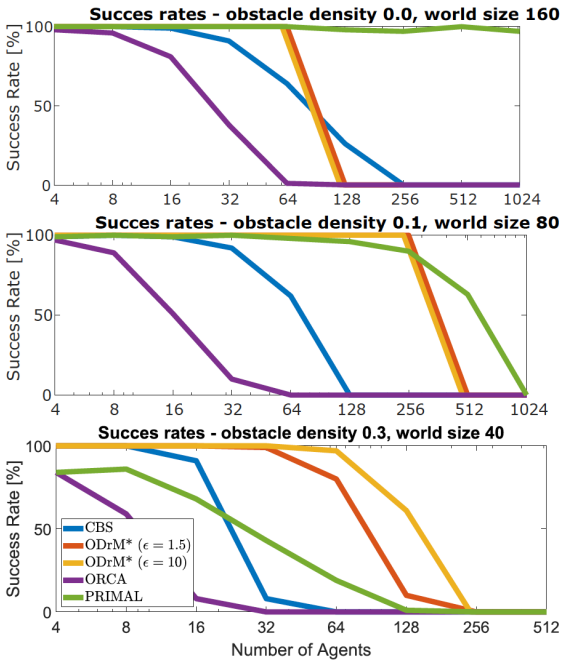

Figure 5: Primal performance in comparison to planners with complete information.
Success rates of the different planners in our three scenarios. PRIMAL outperforms all planners in the top obstacle-free world, slightly outperforms the others in low-obstacle-density worlds, and is strongly outperformed in the high-obstacle-density world. *source: Fig.6 of (Sartoretti et al. 2019a)*

For a purpose of this paper, the last two parts are potentially interesting. Target searching is trained by reinforcement learning. Inspired by PRIMAL and other following papers, the algorithm uses A3C, as it converges quicker than DQN (Mnih et al. 2013) used in the previous version of this algorithm. Navigation is divided into two problems – global navigation to the target and local obstacle avoidance. Avoidance is also solved by RL, specifically by a dual-stream Q-learning and global navigation by DQN.

Results Authors run three different experiments – two simulations and one experiment in a pool. Simulation experiments focused on single- and multi-AUV setting. Both experiments showed that the algorithm is able to find all targets and also has quite a good collision avoidance. The performance was measured in terms of success rate and average path length and compared with three other heuristic approaches for target searching and won. As can be seen from figures 9 and 10, the presented algorithm can find shorter paths with a better success rate than all three other algorithms.

Communication between agents

Information that an agent has at the time of decision can describe the whole world, his focus of view, or can be enriched by information from other agents. If we decide to use

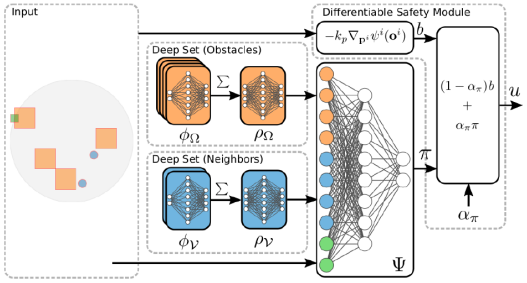

Figure 6: Neural network used in GLAS algorithm. As agent is supposed to treat other agent differently from obstacles, there are two separate networks for each of this types. An input of the network are obstacles and neighbours in some given radius around the agent.
source: Figure 2 of (Rivière et al. 2020)

communication as another source of information for the decision, there is still a lot to do for effective implementation. This section is dedicated to the exploration of recent trends. What information should be shared, with which agents, how to use the obtained information for improving decisions, and so on –these are questions with no simple answer, but all can be a subject of learning.

Communication between agents faces the same dimensionality problem as the whole MAPF problem. With growing message size (in bits), the number of possibilities for one agent in one step also grows. If potential message length is enlarged by one bit, the size of a set containing all possible messages grows twice. Generally spoken, for space with message size n, space size is 2^n. Therefore, message space size grows exponentially with the number of message bits, which makes policy learning difficult, as state-action-message space is growing too much.

(Freed, Sartoretti, and Choset 2020) **Motivation** In this paper, the goal is a bitwise message policy parametrization, which aims to solve the problem with the exponential growth of message space and also presents a policy gradient estimator capable of treating the message part of the gradient, stay unbiased and decrease the variance compared to "typical gradient estimators". This helps to create a more robust solution with faster convergence.

World definition and available information The problem is simplified as follows: every message has just one recipient and a received message in time t influences just an action selected in time $t+1$. Similarly, as in previous papers, agents live in a grid world. The difference is that agents do not know their own goals. Input information in every step is the robot's position and its neighbor's goal, which encourages the agent to communicate to learn its goal.

Technologies The output of this algorithm is a policy that in each step chooses a message to be sent and its probability. This policy is represented as a set of probabilistic distributions per each message bit. Each bit is therefore generated independently, so adding a new message bit causes just linear growth (one new output must be added). This does not

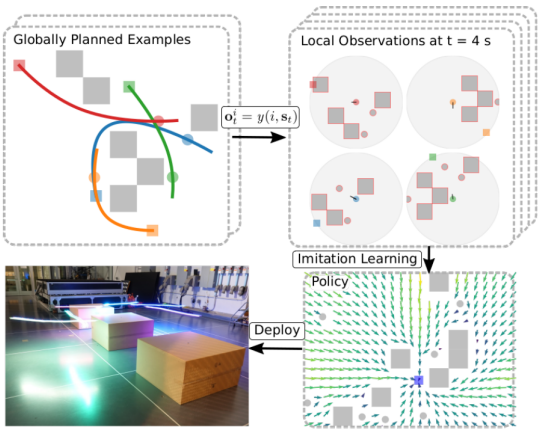

Figure 7: Learning data are generated as follows:
1) generating random map with obstacles, start and end positions for some robots and expert trajectories generated by good planner
2) masking all information which an agent can not see in the real world
3) generate *observation-trajectory* pair from data above
source: Figure 1 of (Rivière et al. 2020)

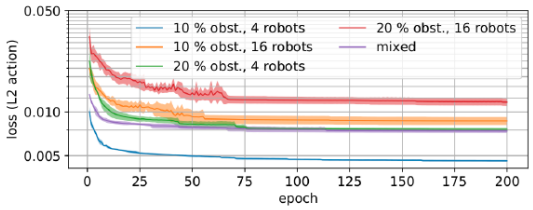

Figure 8: Success rate and control effort with varying numbers of robots in a 8mx8m space. The shaded area around the lines denotes a standard deviation over 5 repetitions. The shaded gray box highlights validation outside the training domain.
source: Fig.7 of (Rivière et al. 2020)

represent well dependencies between message bits, but it is more efficient than generating messages from the joint space of all bits. In theory, such a bitwise stochastic policy can represent any deterministic message policy. The author assumes that the optimal policy is deterministic, so this approach is in their opinion able to learn effective messaging.

Training Two networks are trained – one for the value function and one for policy. The input of the policy network is the agent's observations and a message from the previous step. Value network has different inputs – all environment information, other agent's action in the actual step, and in the case of standard gradient estimation also a message from the previous step. Both networks have similar architecture.

Results This paper presents two theoretical results – new gradient estimator for the value function and policy gradient which includes messaging. This is allowed by simplifications stated above, but the paper proposes the possibility

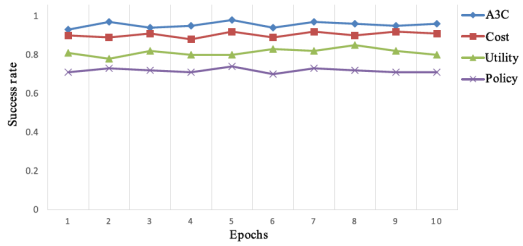

Figure 9: Comparison of A3C and three previous heuristic algorithms (cost, utility and policy).
source: Fig.9 of (Cao, Sun, and Yan 2019)

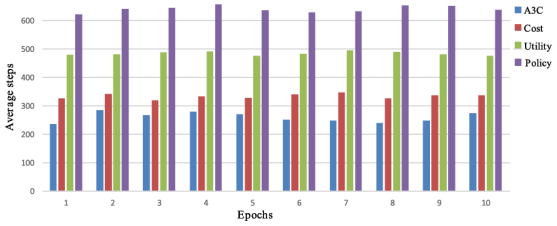

Figure 10: Comparison of A3C and three previous heuristic algorithms (cost, utility and policy).
source: Fig.10 of (Cao, Sun, and Yan 2019)

of generalization into more message receivers.

Experiments were performed on two real physical agents (robots) in a grid world. Possible actions are: left, right, forward, back, not move and are combined with choosing 0/1 for every bit in the message. The episode ends once agents are on their goals or after 256 steps.

Experiments showed, that message encoding together with the newly proposed gradient improves the mean episode reward and speeds up the achievement of the goal (see Figure 11). This shows that better information sharing can be useful for agent's cooperation and learning.

As showed in (Freed, Sartoretti, and Choset 2020), utilization of the potential of received information can also be tricky if the information is not selected properly. In one of the experiment settings, where they used a novel approach for selecting messages, but "old" gradient update, the learner was not able to perform significantly better than a brute-force search, so according to the authors, the algorithm is not able to communicate effectively (share relevant information or use obtained information for better action selection).

(Li et al. 2019) **Motivation** The previously mentioned paper assumed communication with just one (closest) agent. Paper does not deal with message recipients selection and it is mentioned as included in ongoing work. Meanwhile, the second paper to be mentioned (Li et al. 2019) also deals with addressees selection.

World definition and available information Each agent knows information from its FOV and the position of its goal. Every step, each agent must decide its next action. The main

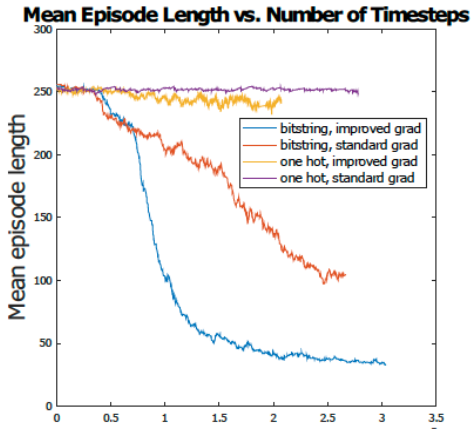

Figure 11: Figure presenting results of experiments authors made. Best results are obtained when using both improvements (one-hot encoding and an improved gradient) together. A higher value is better because it indicates, that episode does not end prematurely before achieving goals. *Fig.3 of (Freed, Sartoretti, and Choset 2020)*

goal is to learn two things – what information to share and with whom. In contrast to the previous paper (Freed, Sartoretti, and Choset 2020) where communication was needed to know its own goal, in this case, the agent knows its goal, but communication can help him to move more effectively.

Technologies For selecting message receivers algorithm uses graph neural networks (GNN). GNN can be in shortcut viewed as a modification of convolutional neural networks that are applicable to graphs.

Filtering information is addressed by convolutional neural networks (CNN). CNN could be interpreted as a feature extractor and its inputs are information from the agent's FOV. Choosing whom to send a message is solved by GNN by creating a matrix of communication graph. In such graph, each node is one agent and the edges between them represent communication channels. In the result, the agent is communicating only with agents which are connected by an edge and are located within some radius, which is one of the hyperparameters. Step action is then chosen by multi-layer perceptron (MLP). MLP has an output of GNN as its input and the same MLP is used by all agents (more specifically, every agent has a copy of MLP with the same weights).

Training The training uses imitation learning (IL), specifically learning from trajectories generated by Conflict-Based Search (Sharon et al. 2015) on 30,000 generated cases (map with obstacles, goals and agents). If a selected action leads to a node or edge collision, such action is replaced by idle action (the agent does not move). Because some episodes can end with deadlock, some of those episodes are given to an expert planner. The resulting trajectory which starts in a deadlock position and ends with a successful solution (from expert planner) became newly a part of training data, so this solution can be learned in the future.

Results Presented results proved that this solution is faster and more stable for a larger bunch of robots than classical gradient estimator and most importantly is able to generalize to different numbers of agents. To be exact, the scalability depends strongly on the agent count for which the network was trained. After proper training, it works nicely for cases with the smaller or the same amount of agents. Also, models trained on greater groups are better on unseen data in general.

Nature inspired path finding

Another widely used approach to many AI problems is evolutionary algorithms and other techniques inspired by a nature. MAPF is not an exception and two recent algorithms are described in this section.

ACO One of algorithms inspired by nature is *Ant Colony Optimization* (ACO) (Dorigo, Birattari, and Stutzle 2006). It is an algorithm inspired by ants' behavior while searching for food.

Simulated ants move in the space and search for a goal and "secrete pheromones" while moving, which causes other ants to follow the path with higher probability. Because this can lead to local optima stuck or even into finding a path to the goal which was randomly selected at first, a concept of evaporation is introduced. It causes pheromones to fade out over time. If the selected path was short, it is likely, that ants can go to the goal and back faster and thus keep the pheromones levels on the path high despite evaporation.

A solution to multi-agent pathfinding via ACO is offered in (Zheng, Wang, and Xi 2018). First, this paper offers some improvements in the path-finding ACO algorithm itself, which leads to secure and quicker convergence. Then the multi-agent nature of the problem is considered by solving possible deadlocks and collisions and the speed of agents is also taken into account. In more detail, each agent's path is changed dynamically according to occurrences of collisions in the given radius around the agent (see figure 12).

In the beginning, the shortest path is found by ACO. Then while robots are moving towards their goals, they can block other agents and this is solved by the firework algorithm, which is able to find the next promising position. Path change (because of collision) for one agent can start chain of changes for other agents. The algorithm tries to avoid deadlocks and struggling in a local optima by incorporating the fireworks algorithm (Tan, Shi, and Tan 2010). For whole algorithm summary, see figure 13. Results show improvement in the average time and path lengths. Unfortunately, paper experiments are restricted to comparison with traditional ACO and (Qu, Huang, and Ke 2015), so it is not possible to assess the results in comparison with classical planning results.

GA Genetic algorithms (Mitchell 1998) can also be applied to MAPF problem. One of the most recent papers is *A method for real-time dynamic fleet mission planning for autonomous mining* (Wahde, Bellone, and Torabi 2019). GA are here used to effectively generate a conflict-free path also with a schedule for a fleet of vehicles in the mine. It also explored the dynamic version, where goals are dynamically assigned to the vehicle after one mission is completed.

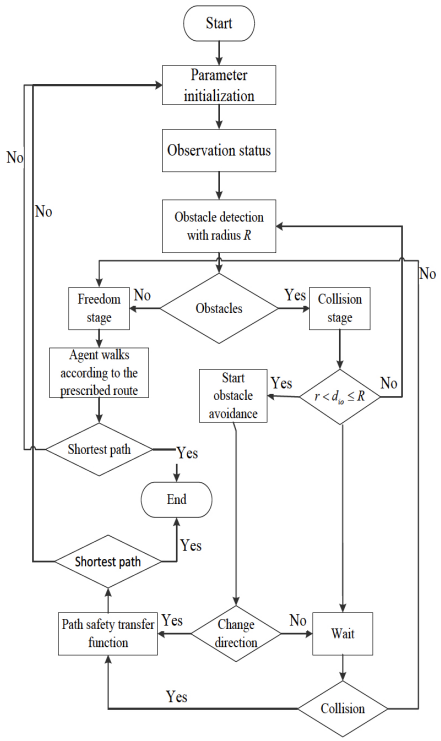

Figure 12: Obstacle avoidance scheme.
source: Fig.1 of (Zheng, Wang, and Xi 2018)

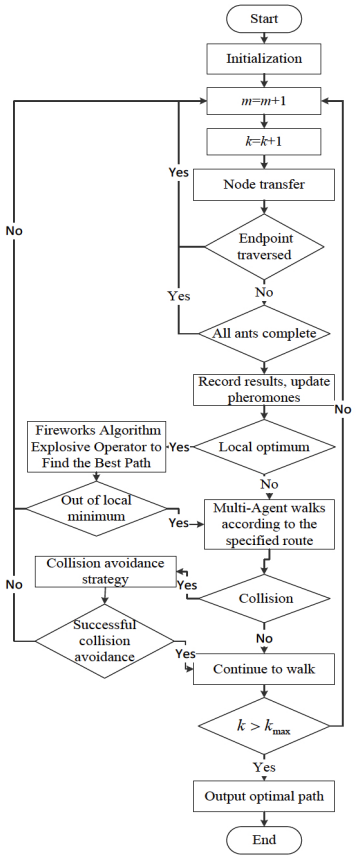

Figure 13: Schema of whole decision algorithm. *source: Fig.2 of (Zheng, Wang, and Xi 2018)*

Dealing with motion in the real world

Motivation Possible usage in MAPF can be not just in theoretical path planning but also in an application to embodied agents (robots). One of the main challenges for robots are moving obstacles (e.g. humans or other robots). When other robots are part of the same team and communication is allowed or the policy is centrally planed, their moves are predictable to some extent, which definitely can not be said about humans. Despite the difficulty of predicting other autonomous entities, modeling their behavior is one of the frequently used approaches. As none of the previously mentioned papers deals with this problem, the following paragraph will shortly present one possible solution.

Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning (Everett, Chen, and How 2018) presents a possible approach. The main advantage of the newly presented algorithm is its ability to deal with a previously unspecified number of movable obstacles. This feature is really useful because the goal is to create robots able to move e.g. in the office, where an amount of pedestrians is variable and potentially large.

World definition and available information The goal is to train policy and every discrete time step decide what to do according to this policy. Action space is divided into 11 discrete actions (combinations of direction and velocity). The observable part of the environment state is the agent's position, velocity, nearest agents (in a given radius), and surrounding obstacles in the form of pictures or video frames. Authors however propose to apply feature extraction to distinguish between different types of obstacles, so that furniture and people are treated differently.

Technologies The ability to deal with the changing amount of obstacles is achieved by using LSTM cells in the neural network. LSTM cells are frequently used in linguistics for a series of words processing. Because every sentence has a different length, LSTMs are designed to deal with variable input lengths. This property is used in this case to create input to the following parts of the network from a bunch of encoded information about obstacles around. In contrast to the majority of papers dealing with movable obstacles, the algorithm presented by (Everett, Chen, and How 2018) does not have any models of obstacles' behavior like assuming people will not change their direction and velocity for some time. The algorithm learns the proper reaction without any modeling.

This algorithm is based on reinforcement learning, specifically on A3C algorithm.

Training decision-maker was trained on a data set of 500 random scenarios and this dataset can be reused for a benchmark for future works.

Results Performance differs in dependency on the number of agents. For 6 or more agents, the newly proposed so-

lution is better than previous work on this topic (Chen et al. 2017). This new algorithm is better in execution time as well as in a success ratio.

Future Work Proposals

In the previous section, various approaches to reliable path planning with the cooperation and collision avoidance were presented. Some of them are still on the start of their path to algorithms usable in the real world and some, like PRIMAL, have big potential to be a good solution to many problems. Almost all authors also suggest improvements as part of result discussion.

PRIMAL itself was improved by its very authors (Sartoretti, Koenig, and Choset 2019). The new algorithm aims to merge the advantages of PRIMAL with a centralized planner to improve the performance in very dense environments (many obstacles or robots). The previous implementation was outperformed by classical planners in this kind of world, but PRIMAL was still able to lead effectively many robots to their goals before stuck. They offer a simple solution – After a fixed number of PRIMAL steps, if agents are not on their goals yet, a centralized planner (ODrM*) has 5 seconds long chance to plan the rest of the moves. It showed reasonable improvement in the case of a small and middle environment with a density of up to 30% and up to 64 agents. For larger groups of agents, this approach is outperformed by centralized planners. The authors also propose future improvement by inventing more clever struggle detection than just a number of steps.

(Li et al. 2019) is a promising communication algorithm, but the policy is learned via multilayer perceptron and conflicts are solved poorly. If the action leads to the conflict, it is just replaced by idle action. Imitation learning is not so helpful for learning conflict-free policy, because the central planner typically prevents collision a few steps ahead, so there are no training data with "bad" situations (e.g. conflict in next action, need to leave the goal and let another agent pass). Collision avoidance is, on the contrary, addressed in PRIMAL, motion planner, and improved ACO, where communication could improve results, because more information could lead to better decisions.

Conclusion

Presented papers cover different approaches and have different goals - shortest paths, collision avoidance, communication. Of course, these objectives are mixed together in all papers, but there is not focus on all of them in one algorithm. A common underlying thing for reinforcement learning is the usage of A3C algorithm, while LSTM for an arbitrary number of obstacles and graphs for communication are quite interesting method transfers.

The main suggestion of this work is to combine suitable collision avoidance module (preferably one with an ability to deal with non-static obstacles), quick and good decentralized policy learner with time- and space-effective communication, which can lead to the decision-maker applicable in (not only) commercial use. This paper presented an overview of promising starting points for this "ideal" decision-maker.

References

Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* (5):834–846.

Cao, X.; Sun, C.; and Yan, M. 2019. Target search control of AUV in underwater environment with deep reinforcement learning. 7:96549–96559. Conference Name: IEEE Access.

Chen, Y. F.; Everett, M.; Liu, M.; and How, J. P. 2017. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1343–1350. IEEE.

Dorigo, M.; Birattari, M.; and Stutzle, T. 2006. Ant colony optimization. *IEEE computational intelligence magazine* 1(4):28–39.

Everett, M.; Chen, Y. F.; and How, J. P. 2018. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3052–3059. IEEE.

Freed, B.; Sartoretti, G.; and Choset, H. 2020. Simultaneous policy and discrete communication learning for multi-agent cooperation. *IEEE Robotics and Automation Letters* 5(2):2498–2505.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Li, Q.; Gama, F.; Ribeiro, A.; and Prorok, A. 2019. Graph neural networks for decentralized multi-robot path planning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*.

Mitchell, M. 1998. *An introduction to genetic algorithms*. MIT press.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Qu, H.; Huang, L.-W.; and Ke, X. 2015. Research of improved ant colony based robot path planning under dynamic environment. *Dianzi Keji Daxue Xuebao/Journal of the University of Electronic Science and Technology of China* 44:260–265.

Rivière, B.; Hönig, W.; Yue, Y.; and Chung, S.-J. 2020. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning. *IEEE Robotics and Automation Letters* 5(3):4249–4256.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T. S.; Koenig, S.; and Choset, H. 2019a. Primal: Pathfinding

via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters* 4(3):2378–2385.

Sartoretti, G.; Wu, Y.; Paivine, W.; Kumar, T. S.; Koenig, S.; and Choset, H. 2019b. Distributed reinforcement learning for multi-robot decentralized collective construction. In *Distributed Autonomous Robotic Systems*. Springer. 35–49.

Sartoretti, G.; Koenig, S.; and Choset, H. 2019. A combined learning- and search-based approach to complete multi-agent path finding. *Proceedings of the IJCAI Workshop on Multi-Agent Path Finding, pages (in print), 2019* 3.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219:40–66.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. S.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tan, Y.; Shi, Y.; and Tan, K. 2010. In *Advances in Swarm Intelligence: First International Conference, ICSI 2010, Beijing, China, June 12-15, 2010, Proceedings, Part I*, 355–364.

Van Den Berg, J.; Guy, S. J.; Lin, M.; and Manocha, D. 2011. Reciprocal n-body collision avoidance. In *Robotics research*. Springer. 3–19.

Wagner, G., and Choset, H. 2015. Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219:1–24.

Wahde, M.; Bellone, M.; and Torabi, S. 2019. A method for real-time dynamic fleet mission planning for autonomous mining. *Autonomous Agents and Multi-Agent Systems* 33(5):564–590.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep sets. In *Advances in neural information processing systems*, 3391–3401.

Zheng, Y.; Wang, L.; and Xi, P. 2018. Improved ant colony algorithm for multi-agent path planning in dynamic environment. In *2018 International Conference on Sensing,Diagnostics, Prognostics, and Control (SDPC)*, 732–737.