ISSN 1064-5624, Doklady Mathematics, 2022. © The Author(s), 2022. This article is an open access publication.
Russian Text © The Author(s), 2022, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2022, Vol. 508, pp. 88–93.

ADVANCED STUDIES IN ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Planning and Learning in Multi-Agent Path Finding

K. S. Yakovlev[a,b,*], A. A. Andreychuk[a], A. A. Skrynnik[a], and A. I. Panov[a,b]
Presented by Academician of the RAS A.L. Semenov

Received October 28, 2022; revised October 31, 2022; accepted November 3, 2022

Abstract—Multi-agent path finding arises, on the one hand, in numerous applied areas. A classical example is automated warehouses with a large number of mobile goods-sorting robots operating simultaneously. On the other hand, for this problem, there are no universal solution methods that simultaneously satisfy numerous (often contradictory) requirements. Examples of such criteria are a guarantee of finding optimal solutions, high-speed operation, the possibility of operation in partially observable environments, etc. This paper provides a survey of modern methods for multi-agent path finding. Special attention is given to various settings of the problem. The differences and between learnable and nonlearnable solution methods and their applicability are discussed. Experimental programming environments necessary for implementing learnable approaches are analyzed separately.

Keywords: path planning, heuristic search, reinforcement learning, multi-agent systems
DOI: 10.1134/S1064562422060229

1. INTRODUCTION

In the general form, the problem of multi-agent path finding is stated as follows. A group of mobile agents (e.g., mobile robots or virtual persons) operates in common space. Each agent is to move to a known goal position, avoiding collisions with the other agents or static and stochastic obstacles. Recently, interest in methods for solving this problem has increased significantly, mainly due to their applications in warehouse and service robotics [1] and in intelligent transportation systems [2].

Various assumptions made at the stage of problem formalization have a significant effect on the choice of solution methods. For example, one of the most widespread and actively studied formalizations is classical multi-agent path finding (classical MAPF). In this problem, it is assumed that there is a centralized controller that has complete information on the state of the environment and all agents (full observability). Time is assumed to be discrete, i.e., at every time step, an agent can perform a single action, namely, a move action or a wait action. The space is discretized in the form of a graph, i.e., the agents are assumed to move only along edges of an a priori given graph and to perform wait actions only at its nodes. Four-connected graphs (grids) are usually used in practice [3]. There are numerous variations of this graph-based centralized setting of the problem. For example, a variant in which the agents' goals are not fixed, i.e., the distribution of agents over goal positions is part of the problem solution is considered in [4]. In [5] it is assumed that each agent can have several goals and has to visit them sequentially. A lifelong problem is considered in [6], namely, after an agent reaches its goal, it is immediately assigned another (previously unknown) goal. Overall, despite the differences in formulation details, centralized variants of multi-agent path finding are usually solved by applying classical nonlearnable algorithms based either on heuristic search in the state space (in some form) [8–10] or on the reduction of the original problem to classical ones in computer science, for example, to the satisfiability of Boolean formulas (SAT) [11] or a network flow problem [12].

In addition to multi-agent path finding problems that assume full observability and centralized control, of interest, including in applications, is an alternative setting in which there is no centralized controller and agents can observe the environment (including other agents) only within a certain radius around them (so-called partial observability). This problem is reasonably formalized in the form of sequential decision making, when at every time step each agent choses to perform a single action relying on the current observation (and, possibly, on the history of observations and interactions with the environment). Reasonably, the problem in this setting is solved by applying reinforcement learning methods [13].

[a] *Artificial Intelligence Research Institute, Moscow, Russia*
[b] *Federal Research Center "Computer Science and Control," Russian Academy of Sciences, Moscow, Russia*
**e-mail: Yakovlev@airi.net*

In what follows, methods for solving both classes of multi-agent path finding problems are considered in more detail.

2. NONLEARNABLE (CLASSICAL) METHODS

Nonlearnable methods for multi-agent path finding are usually used in the case of full observability, a centralized controller, and graph discretization of the agents' working space. The task is to construct a set of non-conflicting trajectories, namely, paths on a graph, including possible wait actions at vertices. It is well known that, on the one hand, in the case of an undirected graph, this problem can be solved in polynomial time [14]. On the other hand, obtaining optimal solutions is NP-hard [15]. If the graph is directed, then even obtaining a nonoptimal solution is an NP-hard problem [16].

There are solution methods based on reducing this problem to other well-known problems in computer science. For example, multi-agent path finding (MAPF) is reduced to SAT in [11], to an integer programming problem in [17], and to a network flow problem in [12]. Among these methods, more widespread are those reducing MAPF to SAT. Likely, the cause is that numerous efficient solvers are available for SAT; as a result, the speed of solving the original problem is also fairly high. The following analogy is worth mentioning. Under certain assumptions, MAPF can be treated as a 15 puzzle game. This approach is used in modern algorithms, for example, in Push and Rotate [18], designed for fast obtaining nonoptimal solutions.

Another approach to the solution of MAPF is based on algorithms involving direct search on a graph. Obviously, heuristic versions of search are used to improve its efficiency. A classical heuristic search algorithm is A* from [7]. With certain modifications, it can be used to find optimal MAPF solutions [8], but overall this approach is not very efficient, since, in fact, it treats all agents as a single meta-agent and carries out search in a combined space with a branching coefficient depending exponentially on the number of agents. To avoid a combinatorial explosion, various decoupled search techniques are applied. Examples are the algorithms CBS [9] and M* [10]. Both guarantee the optimality of found solutions and have a variety of modifications, including ones aimed at improving computational efficiency, while preserving the optimality of the solution [19, 20]; modifications that trade off optimality against computational efficiency [21]; and modifications solving MAPF under milder constraints, for example, in continuous time [22].

Another approach to MAPF solution based on heuristic search is prioritized planning [23]. In this case, each agent is assigned a priority and then only individual paths are sought. All earlier planned trajectories are considered unchangeable (in other words, dynamic obstacles for the current agent). Theoretically, this approach does not guarantee optimality; moreover, it does not even guarantee that the solution of the problem will be found if it exists. Nevertheless, such a guarantee can be given for a certain class of problems [24]. Moreover, in practice, prioritized algorithms find close-to-optimal solutions in numerous instances, while spending much less computational resources. That is why algorithms of this class are often used in robotics [25].

3. LEARNABLE METHODS

There are several variants of using machine learning methods in the context of MAPF with full observability and a centralized controller. First, these methods can be used to select an MAPF algorithm most suitable for a particular problem (map, positions of agents) [26, 27]. Second, machine learning methods can be used to learn various heuristic selection rules involved in classical MAPF-solving algorithms [28, 29]. In recent years, reinforcement learning methods have become widespread. They are able to solve MAPF in decentralized and partially observed settings. One of the first works in this direction was [30], where a learning strategy called PRIMAL was presented. Later, it was improved and generalized to lifelong search [31], when after reaching its goal, an agent does not finish the current episode, but receives a new task. Both algorithms used demonstration trajectories generated by the ODrM* search algorithm [32]. Algorithms of the PRIMAL family use a complicated reward function and make a large number of partial assumptions concerning specific conditions and maps (domain knowledge), for example, additional penalty for conflicts or the assumption that a local observation includes not only positions of other agents, but also their goals. Similar assumptions were used in [33], which proposes another learnable algorithm for MAPF, but in the case of more complicated dynamic models of agents (e.g., such as quadcopters). Learnable methods that use complete information on static elements of the environment (global information on the positions of other agents is not available to them) were proposed in [34, 35].

In addition to algorithms developed specially for MAPF, there are universal approaches of multi-agent reinforcement learning that can be used to solve MAPF. Among the wide variety of classical algorithms for single-agent learning (which is called independent), as well-established one for partially observable multi-agent problems is the policy gradient approach, a popular implementation of which is known as proximal policy optimization (PPO) [36–38]. Another direction is centralized training in cooperative policies. Algorithms of this type are usually trained in a centralized fashion, using global information on the environment, while their testing is decentralized. For example, QMIX [39] uses hyper-networks for training

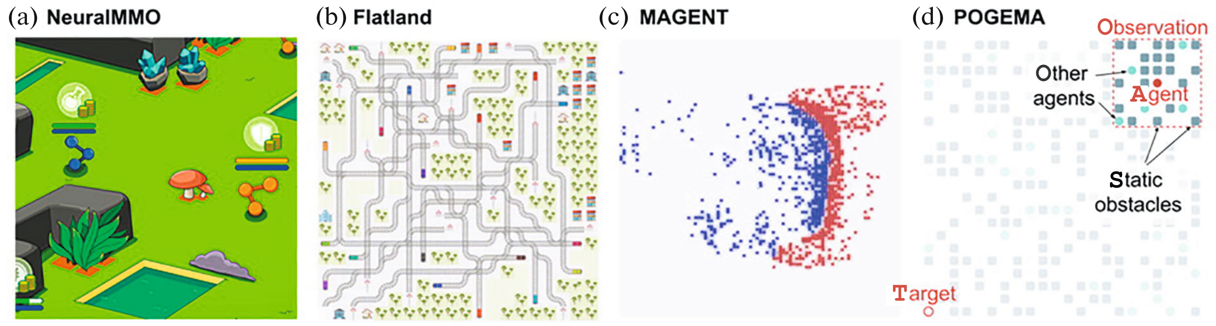

Fig. 1. Examples of environments used in learnable methods for solving multi-agent problems.

individual policies via a mixing utility network. In training, each network receives only a local observation, and it is optimized by a hyper-network, to which the global state is available. The learning algorithms MADDPG [40] (off-policy learning) and MAPPO [41] (on-policy learning), the critic uses a centralized network. This is a general learning approach when the critic uses the global state of the environment for better training a utility function approximator. A policy-determining actor receives, as input, only a partial observation, but implicitly uses a general observation, using critic's estimates. The FACMAC algorithm [42] is a combination of MADDPG and QMIX, so it can be used for both discrete actions, using the Gumbel–Softmax trick, and for continuous actions.

MARL algorithms are rather strongly optimized for a number of environments, which have become classic for testing, for example, SMAC [43], which uses the StarCraft 2 game. In contrast to single-agent reinforcement learning, ready-for-use implementations in open access are much fewer, and available ones are suitable only for rather simple problems. The main cause is that they represent slow implementations without parallelization intended for only several millions of steps in the environment and for simple fully connected architectures as approximators. As a result, most researchers prefer using well-known decentralized approaches. However, this leads to another extreme, namely, the proposed algorithms exploit subject area knowledge, which limits their applicability to a broad class of MAPF problems.

A promising direction in the development of more advanced methods for MAPF can be model-based reinforcement learning [44]. Prediction of the other agents' policies and allowance for this model in the construction of its own agent policy can be especially useful in a heterogeneous group of agents, where each agent can have its own policy [45]. Another open niche in MARL is the use of demonstrations in learning. Indeed, demonstrations can significantly accelerate the learning process and can also allow using modern transformer and diffusion models. This is especially important for MAPF problems, for which there are strong planning algorithms.

4. EXPERIMENTAL ENVIRONMENTS FOR TESTING ALGORITHMS

Experimental online environments are hardly used in nonlearnable approaches to the solution of MAPF, since these approaches do not assume learning via the interaction with the environment. An opposite situation occurs in the reinforcement learning community, where there are numerous environments, but most of them are intended for games and are characterized by numerous additional features that have nothing to do with MAPF (e.g., stocks, counteracting opponents, etc.) (see Fig. 1).

An example of a game environment is NeuralMMO [46], which is a simplified version of a multiplayer network game with a group of agents solving the task of survival and resource accumulation. A team of eight agents competes with other 15 teams on a procedurally generated map of 128 by 128 cells. The environment is partially observable, but the agents can communicate with each other. Although this problem is rather complicated, it is far from practical application and requires mainly reactive choices of actions based on a set of rules, rather than planning or path finding.

A well-known environment designed specifically for MAPF is Flatland [48], which is a simplified, yet realistic environment for scheduling railway networks. Here, agents are trains, which are to move from one station to another in a single-track railway, avoiding conflicts with each other. Within the framework of this problem, several competitions have been conducted in order to research reinforcement learning algorithms. However, it turned out that access to the full state of the environment provides a significant advantage for planning and replanning approaches [49]. Another shortcoming of this environment is that it works very slowly (near 200 steps per second for small maps) in the regime of observations intended for learnable algorithms.

MAGENT is a set of environments from the PettingZoo library [47]. It is designed for modeling the role behavior of agents capable of moving from one place to another and interacting with each other in various ways. The implementation in C++ significantly accelerates the interaction, but this environment possesses a limited set of scenarios (map types) and has no interface for testing solutions based on approaches other than reinforcement learning.

The most suitable environment for MAPF problems is POGEMA [50], which was specially designed for problems in partially observable setting on cellular maps. The authors emphasize that agents receive information only from a bounded space around them and cannot transfer information to each other, which considerably complicates the problem for both planning and learnable algorithms. The main advantages of this environment are its flexibility and the performance speed. POGEMA allows for any user-created maps of obstacles and supports three regimes determined by what happens after an agent reaches its goal: the agent receives a new goal (lifelong path search), agents that disappear (after reaching goals), and agents that do not disappear until the end of an episode.

5. CONCLUSIONS

In recent years, methods for MAPF have been actively developed as motivated by their applications in various practical areas (warehouse robotics, transportation systems, etc.). Under centralized control and full observability, the solution approach usually involves nonlearnable methods based on heuristic search or on reducing MAPF to other classical problems in computer science (SAT, network flow, etc.). In the case when centralized control is absent and/or full information on the environment is not available to agents, one often applies reinforcement learning methods based on either adaptation of well-known search strategies for individual agents or on the "centralized training–decentralized execution" paradigm. In our view, the most promising (and least studied) would be a combined approach involving both reinforcement learning and classical planning methods (heuristic search, etc.).

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

OPEN ACCESS

REFERENCES

1. H. Ma and S. Koenig, "AI buzzwords explained: Multi-agent path finding (MAPF)," AI Matters **3** (3), 15–19 (2017).
2. R. Morris, C. S. Păsăreanu, K. Luckow, W. Malik, H. Ma, T. K. Satish Kumar, and S. Koenig, "Planning, scheduling and monitoring for airport surface operations," *Workshops at the 30th AAAI Conference on Artificial Intelligence* (2016).
3. P. Yap, "Grid-based path-finding," *Conference of the Canadian Society for Computational Studies of Intelligence* (Springer, Berlin, 2002), pp. 44–55.
4. H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems* (2016), pp. 1144–1152.
5. M. Liu, H. Ma, J. Li, and S. Koenig, "Task and Path planning for multi-agent pickup and delivery," *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems* (2019), pp. 1152–1160.
6. J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. Satish Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (2021), pp. 11272–11281.
7. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Trans. Syst. Sci. Cybern. **4** (2), 100–107 (1968).
8. T. Standley, "Finding optimal solutions to cooperative pathfinding problems," *Proceedings of the 24th AAAI Conference on Artificial Intelligence* (2010), pp. 173–178.
9. G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," Artif. Intell. **219**, 40–66 (2015).
10. G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 3260–3267.
11. P. Surynek, A. Felner, R. Stern, and E. Boyarski, "Efficient SAT approach to multi-agent path finding under the sum of costs objective," *Proceedings of the 22nd European Conference on Artificial Intelligence* (2016), pp. 810–818.
12. J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," *Algorithmic Foundations of Robotics X* (Springer, Berlin, 2013), pp. 157–173.
13. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. (Bradford Books, 2018).

14. D. Kornhauser, G. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," *The 25th Annual Symposium on Foundations of Computer Science* (1984), pp. 241–250.
15. D. Ratner and M. Warmuth, "The ($n^2 - 1$)-puzzle and related relocation problems," J. Symb. Comput. **10** (2), 111–137 (1990).
16. B. Nebel, "On the computational complexity of multi-agent pathfinding on directed graphs," *Proceedings of the 20th International Conference on Automated Planning and Scheduling* (2020), pp. 212–216.
17. J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," IEEE Trans. Rob. **32** (5), 1163–1177 (2016).
18. B. De Wilde, A. W. Ter Mors, and C. Witteveen, "Push and rotate: A complete multi-agent pathfinding algorithm," J. Artif. Intell. Res. **51**, 443–492 (2014).
19. E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and E. Shimony, "ICBS: Improved conflict-based search algorithm for multi-agent pathfinding," *Proceedings of the 24th International Conference on Artificial Intelligence* (2015), pp. 740–746.
20. J. Li, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Symmetry-breaking constraints for grid-based multi-agent path finding," *Proceedings of the 33rd AAAI Conference on Artificial Intelligence* (2019), pp. 6087–6095.
21. M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," *Proceedings of the 7th Annual Symposium on Combinatorial Search* (2014).
22. A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," Artif. Intell. **305**, 103662 (2022).
23. M. Erdmann and T. Lozano-Perez, "On multiple moving objects," Algorithmica **2** (1), 477–521 (1987).
24. M. Cap, J. Vokrinek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," *Proceedings of the 25th International Conference on Automated Planning and Scheduling* (2015), pp. 324–332.
25. K. Yakovlev and A. Andreychuk, "Any-angle pathfinding for multiple agents based on SIPP algorithm," *Proceedings of the 17th International Conference on Automated Planning and Scheduling* (2017), pp. 586–594.
26. O. Kaduri, E. Boyarski, and R. Stern, "Algorithm selection for optimal multi-agent pathfinding," *Proceedings of the International Conference on Automated Planning and Scheduling* (2020), pp. 161–165.
27. J. Ren, V. Sathiyanarayanan, E. Ewing, B. Senbaslar, and N. Ayanian, "MAPFAST: A deep algorithm selector for multi agent path finding using shortest path embeddings," *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems* (2021), pp. 1055–1063.
28. J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.
29. T. Huang, S. Koenig, and B. Dilkina, "Learning to resolve conflicts for multi-agent path finding with conflict-based search," *Proceedings of the AAAI Conference on Artificial Intelligence,* 2021, Vol. 35, No. 13, pp. 11246–11253.
30. G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. Satish Kumar, S. Koenig, and H. Choset, "PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning," IEEE Rob. Autom. Lett. **4** (3), 2378–2385 (2019).
31. M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning-lifelong," IEEE Rob. Autom. Lett. **6** (2), 2666–2673 (2021).
32. C. Ferner, G. Wagner, and H. Choset, "ODrM* optimal multirobot path planning in low dimensional search spaces," *2013 IEEE International Conference on Robotics and Automation* (2013), pp. 3854–3859.
33. B. Riviere, W. Hönig, Y. Yue, and S. J. Chung, "GLAS: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning," IEEE Rob. Autom. Lett. **5** (3), 4249–4256 (2020).
34. Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "MAPPER: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems* (*IROS*) 2020, pp. 11748–11754.
35. B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," IEEE Rob. Autom. Lett. **5** (4), 6932–6939 (2020).
36. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms (2017). arXiv preprint arXiv:1707.06347
37. A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. Panov, "Pathfinding in stochastic environments: Learning vs planning," PeerJ Comput. Sci. **8**, e1056 (2022).
38. C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, and R. Józefowicz, "Dota 2 with large scale deep reinforcement learning" (2019). arXiv preprint arXiv:1912.06680
39. T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," *International Conference on Machine Learning* (2018), pp. 4295–4304.
40. R. Lowe, Yi Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *31st Conference on Neural Information Processing Systems* (2017).
41. C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative, multi-agent games" (2021). arXiv preprint arXiv:2103.01955
42. B. Peng, T. Rashid, C. Schroeder de Witt, P. A. Kamienny, P. Torr, W. Böhmer, and S. Whiteson, "Facmac: Factored multi-agent centralised policy gradients," Adv. Neural Inf. Process. Syst. **34**, 12208–12221 (2021).

43. M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C. M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge" (2019). arXiv preprint arXiv:1902.04043

44. T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "Model-based reinforcement learning: A survey" (2020). http://arxiv.org/abs/2006.16712

45. A. Skrynnik, Y. Yakovleva, D. Davydov, K. Yakovlev, and A. I. Panov, "Hybrid policy learning for multi-agent pathfinding," IEEE Access **9**, 126034–126047 (2021). https://doi.org/10.1109/ACCESS.2021.3111321

46. J. Suarez, Y. Du, P. Isola, and I. Mordatch, "Neural MMO: A massively multiagent game environment for training and evaluating intelligent agents" (2019). arXiv preprint arXiv:1903.00784

47. J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sulliva, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, and N. Williams, "Pettingzoo: Gym for multi-agent reinforcement learning," Adv. Neural Inf. Process. Syst. **34**, 15032–15043 (2021).

48. F. Laurent, M. Schneider, C. Scheller, J. Watson, J. Li, Z. Chen, Y. Zheng, S. H. Chan, K. Makhnev, O. Svidchenko, and V. Egorov, "Flatland competition 2020: MAPF and MARL for efficient train coordination on a grid world," *NeurIPS 2020 Competition and Demonstration Track* (2021), pp. 275–301.

49. J. Li, Z. Chen, Y. Zheng, S. H. Chan, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, "Scalable rail planning and replanning: Winning the 2020 flatland challenge," *Proceedings of the International Conference on Automated Planning and Scheduling* (2021), Vol. 31, pp. 477–485.

50. A. Skrynnik, A. Andreychuk, K. Yakovlev, and A. I. Panov, "POGEMA: Partially observable grid environment for multiple agents" (2022). arXiv preprint arXiv:2206.10944.

Translated by I. Ruzanova