Implicitly Coordinated Multi-Agent Path Finding under Destination Uncertainty: Success Guarantees and Computational Complexity (Extended Abstract)*

Bernhard Nebel[1†], **Thomas Bolander**[2], **Thorsten Engesser**[1] and **Robert Mattmüller**[1]
[1]Albert-Ludwigs-Universität, Freiburg, Germany
[2]Technical University of Denmark, Lyngby, Denmark
nebel@uni-freiburg.de, tobo@dtu.dk, {engesser, mattmuel}@cs.uni-freiburg.de

Abstract

In multi-agent path finding (MAPF), it is usually assumed that planning is performed centrally and that the destinations of the agents are common knowledge. We will drop both assumptions and analyze under which conditions it can be guaranteed that the agents reach their respective destinations using *implicitly coordinated plans* without communication.

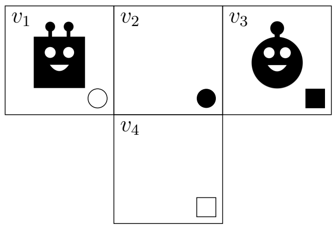

Figure 1: Multi-agent coordination example

1 Introduction

In a spatial multi-agent environment, e.g., a warehouse [Wurman *et al.*, 2008], a street intersection [Dresner and Stone, 2008], an airport [Hatzack and Nebel, 2013], or a video game [Lawrence and Bulitko, 2013], agents have to move to different destinations in a collision-free manner. Such scenarios can be formalized as *multi-agent path finding* (MAPF) problems [Ma and Koenig, 2017].

In its most basic variant, the problem can be described as follows. Given an undirected, simple graph $G = (V, E)$, a set of agents A, an initial configuration assigning agents to distinct vertices, and a goal configuration with another assignment of agents to distinct vertices, the question is how one can transform the initial configuration into the goal configuration by single movements, where one agent moves from a vertex to an empty adjacent vertex.

Often, the graph is given as a grid map as in Figure 1, where agents can move to orthogonally adjacent empty grid cells. In the displayed situation, the circular agent C wants to go to the cell marked by the solid circle and the square agent S wants to reach the place with the solid square (the empty circle and square will only become important later). One could come up with the following movements:

1. C moves to v_2 and then to v_4,
2. S moves to v_2 and then to destination field v_3, and
3. C finally moves to destination field v_2.

This problem and a number of variants have been studied quite extensively, the computational complexity of these problems has been determined [Kornhauser *et al.*, 1984; Ratner and Warmuth, 1986; Surynek, 2010; Yu and LaValle, 2013], and a number of optimal and sub-optimal algorithms have been proposed [Wang and Botea, 2011; Luna and Bekris, 2011; de Wilde *et al.*, 2014; Felner *et al.*, 2017; Botea *et al.*, 2018]. The assumption has usually been that movements are computed centrally before execution starts. Furthermore, because of this assumption, destinations are considered to be common knowledge. Here we drop both assumptions and analyze whether the agents are still able to coordinate their movements in order to reach their respective destinations. Such scenarios are, for instance, plausible in human-robot interactions or when agents do not share a common communication channel.

As a first step, planning as well as execution are assumed to be distributed with no communication between the agents. We call this setting ***distributed MAPF***. In order to cope with the problem that the generated movement plans might be incompatible, replanning might be necessary. The question is then whether it is still possible to guarantee successful executions and what the computational price for such implicitly coordinated executions is. As we show, success guarantees can be given if we assume all agents to be of a certain type.

As a second step, we drop the assumption that destinations are common knowledge. We call the resulting path-finding problem *MAPF under destination uncertainty* or simply ***MAPF/DU***. In order to illustrate this point, let us again consider the situation in Figure 1, but unlike before, let us assume that each agent knows about its own destination with certainty (the solid circle and square), but there is uncertainty

*The full version of this paper has been published as: Bernhard Nebel, Thomas Bolander, Thorsten Engesser and Robert Mattmüller, "Implicitly Coordinated Multi-Agent Path Finding under Destination Uncertainty: Success Guarantees and Computational Complexity," *Journal of Artificial Intelligence Research* **64**: 497–527, 2019.

†Contact Author

about the destinations of the other agent (the empty circle and square are considered as additional potential destinations in addition to the solid circle and square for C and S, respectively).

Here, we first have to come up with a solution concept. We will introduce *strong branching plans* that correspond to *implicitly coordinated policies* as they have been proposed in the area of epistemic planning [Engesser *et al.*, 2017]. We then analyze joint execution of these branching plans as a generalization of joint executions for the fully observable case.

Finally, in the full paper, we have a look at the computational complexity of deciding the existence of bounded strong plans. We show that this problem is PSPACE-complete (in the number of agents). This demonstrates that communication about destinations pays off significantly. For the case that we deal only with few agents, we show in the full paper that deciding existence and bounded existence can be done in time polynomial in the size of the graph provided the number of agents is fixed.

2 Distributed MAPF

While in MAPF one usually considers the generation of a plan by a central instance and leaves the distributed execution to the agents, we now consider the setting where each agent generates a plan—consisting of its own movements and the movements of the other agents, leading to the goal configuration. In the following, we make the assumption that all such plans are cycle-free, i.e., never visit the same global configuration twice. This is, of course, not a restriction since such a cycle could always be removed without invalidating the plan. We call such a plan ***implicitly coordinated*** since the planning agent presupposes that the other agents behave in a cooperative way. The underlying basic assumption is, of course, that all agents want to reach the goal configuration. But it would be a coincidence when all of them came up with the same plan. Nevertheless, if one agent acts, this will follow a plan towards the goal configuration—and so will never end up in a ***dead end***, i.e., a state from which the goal state cannot be reached. And if an action by another agent was not anticipated, then one can replan in order to account for this unanticipated move.

After all agents have planned, we have a ***family of plans***. ***Joint execution*** of this family of plans is then performed in an asynchronous, interleaved fashion. From all the agents that have as their first action one of their own moves, one agent is chosen and its movement is executed. For all the other agents the following happens: Either the movement was anticipated and then the movement is removed from the plan or the agent has to replan from the new situation. The interesting question is, whether such an asynchronous, distributed execution is guaranteed to eventually lead to the desired goal configuration and how many steps it takes to reach the common goal. The ***execution length*** of a joint execution is defined to be the number of steps that are used to reach the goal configuration.

The interesting question is, whether we can find conditions that guarantee success for such joint executions with replanning in the general case. In order to demonstrate one of the issues, let us assume that in our initial example in Fig. 1 both agents come up with plans where the other agent starts with a movement.

We call these plans *lazy* because they put the burden on an agent other than oneself. We say that an agent is ***lazy*** (wrt. a set of plans) if it sometimes generates lazy plans. Clearly, lazy agents can produce plans that can lead to ***deadlock*** situations, i.e., situations in which all agents are waiting for each other to act first. Note that a *deadlock* can occur although none of the plans contain a *dead end*.

We call an agent ***eager*** if it never generates lazy plans (with respect to itself and the given set of plans). With eager agents, we avoid deadlock situations, since there is always at least one agent that can act.

However, simple eager agents (being eager with respect to the set of all plans) have serious problems. The first problem is that they may plan to make moves that increase the distance to the goal configuration, e.g., by moving away from the destination node. The second problem, resulting from the first one, is that joint execution with replanning can easily lead to infinite executions. Note that this can happen even though we have required the plans to by cycle-free. The reason for that is that replanning is always done from the current state not taking into account previously executed actions.

The problem of infinite executions can be addressed by restricting the set of plans Π to *shortest plans*. We call a plan ***optimally eager*** with respect to agent i if it is not lazy with respect to agent i and the set of shortest plans. Agents producing only such plans are called ***optimally eager agents*** [Bolander *et al.*, 2018]. As can be easily shown, such agents are always successful, provided the instance is solvable at all.

While this is good news, the bad news is that this implies that the agents have to solve an NP-hard problem (generating a shortest eager plan) not only once, but potentially after each action execution. Now the question may come up whether a computationally simpler version would be possible. Instead of trying to find the shortest, eager plan in order to avoid infinite executions, one can restrict replanning in a way such that already executed actions have to be part of an updated plan. In other words, when an agent has to replan, it creates the plan from the initial configuration and starts with the already executed actions. [1] Agents that replan in this way are called ***conservative agents***. An agent is a ***conservative eager agent*** if it is eager with respect to the agent and the set of plans containing the already executed plan as a prefix.

This way, one never will create a cyclic execution (since we assumed that plans are never cyclic), hence, executions are always finite. Note that this condition will never lead to a dead end, because an acting agent has always a valid plan to reach the goal configuration. Furthermore, this positive result does not depend on generating shortest plans, so we are not forced to solve NP-hard problems. Unfortunately, however, the polynomial upper bound for the number of movements goes out of the window. The reason is that it is possible that by worst-case decisions on who acts next the eager agents might be forced to visit the entire state space, although a much shorter solution exist.

[1]This is somewhat similar to tabu search [Glover, 1986].

3 MAPF/DU

Let us generalize the MAPF problem to a setting where the agents are only partially informed about the destinations of the other agents. This means that the goal configuration is not common knowledge any longer, but only the agent itself knows its own destination. Common knowledge are the possible destinations for each agent. We, of course, still assume that all agents are cooperative, i.e., that they want to reach the goal configuration.

In order to enable the agents to recognize that the goal configuration has been reached, we now need some limited form of communication. We add a *success announcement* action for each agent. This action can be executed when the agent has reached its destination. Only by using such an action, the agents can establish common knowledge that they all have reached their respective destinations. We require that after the announcement the agent is not allowed to move anymore. However, an agent might visit its true destination without revealing it.

In the original MAPF problem, the state space for the planning process is simply the space of all configurations of the agents in the graph. For the MAPF problem with destination uncertainty we also have to take into account the possible belief states of all the agents. For this reason, we have to make the knowledge about possible destinations part of the state space as well. Furthermore, we will distinguish between the ***objective perspective*** capturing just the agent configuration together with the common knowledge (as seen from the outside) and the ***subjective perspective*** of an agent i, where in addition the actual destination of i is known.

An implicitly coordinated plan is now a plan, where agents need to make ***perspective shifts***, i.e., putting themselves into the shoes of other agents. This means that they have to plan with the knowledge of the acting agent instead of their own knowledge. Furthermore, since they do not know the actual destination of the other agent, they have potentially to *branch on the possible destinations* of the other agent when they make a perspective shift.

In order to illustrate these concepts, let us consider the situation as depicted in Figure 2. Assuming that agent S starts, it could come up with the implicitly coordinated branching plan as depicted in Figure 3. In this plan, S starts by moving from v_1 to v_4. After that, S makes a perspective shift to C (signaled by the marker C: in the plan) and then branches on the possible destinations of C. In each case, it plans for C to move from v_2 to v_1. In the case the destination was v_1, C can announce success (marked by $(C, \mathcal{S})$) and then C can plan

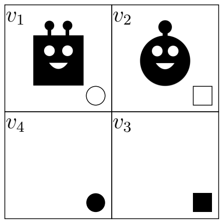

Figure 2: Small example

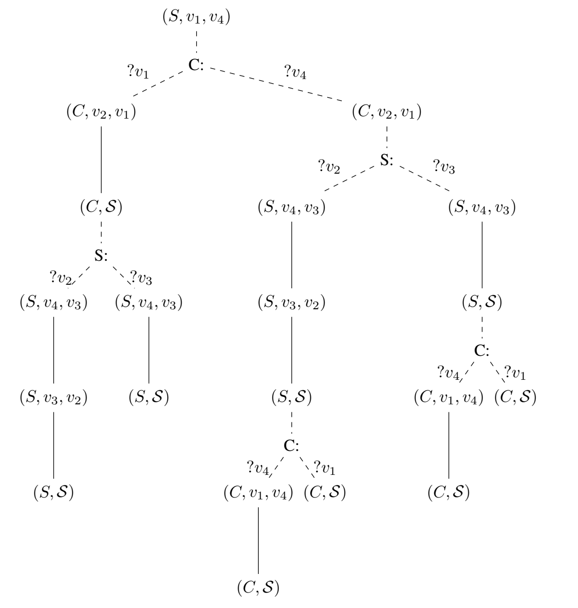

Figure 3: Branching plan

for S to move to its destination.

One of the key results in the original paper is based on the concept of ***stepping stones*** in such branching plans. These are situations in which one agent can reach all possible destinations without waiting for another agent to move out of the way and for all the destinations the situation resulting after having made a success announcement is solvable. In the example shown in Fig. 3, the first perspective shift to C is not happening in a stepping stone situation, because the right branch does not end in a success announcement before making the perspective shift to S. All the other perspective shifts appear in stepping stone situations, however. The key result is now that an implicitly coordinated plan needs to branch only in stepping stone situations, which implies that the depth of such plans can be bounded polynomially.

Joint execution of implicitly coordinated branching plans proceed as in the fully observable case. However, it can happen that only a subset of the agents can generate a plan initially as shown in Fig 4. Here C and S can form an implicitly coordinated plan while the triangle agent T is clueless. By executing the plans, later on the other agents might learn how to successfully complete the plan, however. In our example,

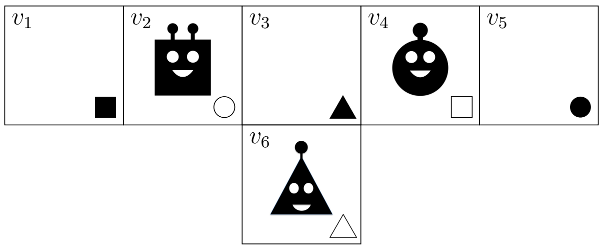

Figure 4: MAPF/DU instance that is only solvable for S and C initially

after S has moved to v_1 announcing success, T can form an implicitly coordinated plan where C moves to its destination, and then T does so. The question of whether there are conditions under which success is guaranteed is much harder to answer than in the fully observable case. It tuns out that it is not enough to require all agents to be *optimally eager agents*. It is possible to come up with an example that leads to cyclic executions. Conservatism is enough, but could again lead to exponentially longer executions. Conservative optimally eager agents, however, are enough to guarantee success and polynomial executions.

Compared with the fully observable case, the computational problem of deciding whether a plan of certain length exists is much harder. It is PSPACE-complete instead of merely NP-complete, demonstrating the value of communication.

4 Related Work

There is a long tradition of analyzing general distributed planning and acting [desJardins *et al.*, 1999]. Brenner and Nebel [2009], for example, looked at this problem. They proposed as one of their benchmarks a problem similar to the MAPF/DU problem. However, their solution is simply the generation of self-interested, greedy plans.

Distributed POMDPs (decPOMPDs) allow for distributed execution [Goldman and Zilberstein, 2004] and might therefore be considered as providing solutions to the problems such as MAPF/DU. However, decPOMDPs are based on a central offline planning process, a problem which can be overcome by interactive POMDPs [Gmytrasiewicz and Doshi, 2005].

The approach that comes closest to the one proposed by us is epistemic multi-agent planning [Löwe *et al.*, 2011; Bolander and Andersen, 2011; Andersen *et al.*, 2012; Muise *et al.*, 2015; Engesser *et al.*, 2017; Bolander *et al.*, 2018]. In fact, the MAPF/DU problem is an instance of epistemic planning. In contrast to the general undecidable problem, we were able to come up with some positive results, though.

Finally, for the related multi-robot path-finding problem there exist some proposals to solve the problem in a truly distributed fashion [van den Berg *et al.*, 2008; Cáp *et al.*, 2015]. However, they tend to be reactive and cannot guarantee completeness as we do or they need some form of minimal communication.

5 Summary

We generalized the MAPF problem to a setting with distributed planning and no communication—*distributed MAPF*—and then further to a setting where the destinations are not common knowledge anymore—*MAPF/DU*. We were able to show that it is possible to guarantee success (provided a central solution exists), if the agents are of a certain type in both cases. On the downside, the computational problem of generating plans became much harder indicating that communication can significantly reduce computational costs.

Acknowledgements

This research was partly supported by DFG as part of the PACMAN project within the HYBRIS research group (NE 623/13-1) and partly by DAAD as part of the Australia-Germany Joint Research Cooperation Scheme 2017/18, project 57319997, and as part of the France-Germany Joint Research Cooperation 2017/18, project 57317892.

References

[Andersen *et al.*, 2012] Mikkel Birkegaard Andersen, Thomas Bolander, and Martin Holm Jensen. Conditional epistemic planning. In *Logics in Artificial Intelligence - 13th European Conference (JELIA-12)*, pages 94–106, 2012.

[Bolander and Andersen, 2011] Thomas Bolander and Mikkel Birkegaard Andersen. Epistemic planning for single and multi-agent systems. *Journal of Applied Non-Classical Logics*, 21(1):9–34, 2011.

[Bolander *et al.*, 2018] Thomas Bolander, Thorsten Engesser, Robert Mattmüller, and Bernhard Nebel. Better eager than lazy? How agent types impact the successfulness of implicit coordination. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixteenth International Conference (KR-18)*, pages 445–453, 2018.

[Botea *et al.*, 2018] Adi Botea, Davide Bonusi, and Pavel Surynek. Solving multi-agent path finding on strongly biconnected digraphs. *J. Artif. Intell. Res.*, 62:273–314, 2018.

[Brenner and Nebel, 2009] Michael Brenner and Bernhard Nebel. Continual planning and acting in dynamic multiagent environments. *Autonomous Agents and Multi-Agent Systems*, 19(3):297–331, 2009.

[Cáp *et al.*, 2015] Michal Cáp, Jirí Vokrínek, and Alexander Kleiner. Complete decentralized method for on-line multirobot trajectory planning in well-formed infrastructures. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS-15*, pages 324–332, 2015.

[de Wilde *et al.*, 2014] Boris de Wilde, Adriaan ter Mors, and Cees Witteveen. Push and rotate: a complete multiagent pathfinding algorithm. *J. Artif. Intell. Res.*, 51:443–492, 2014.

[desJardins *et al.*, 1999] Marie desJardins, Edmund H. Durfee, Charles L. Ortiz, Jr., and Michael Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, 1999.

[Dresner and Stone, 2008] Kurt M. Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *J. Artif. Intell. Res.*, 31:591–656, 2008.

[Engesser *et al.*, 2017] Thorsten Engesser, Thomas Bolander, Robert Mattmüller, and Bernhard Nebel. Cooperative epistemic multi-agent planning for implicit coordination. In *Proceedings of the Ninth Workshop on Methods for Modalities (M4MICLA-17)*, pages 75–90, 2017.

[Felner *et al.*, 2017] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan R. Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Proceedings of the Tenth International Symposium on Combinatorial Search (SOCS-17)*, pages 29–37, 2017.

[Glover, 1986] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & OR*, 13(5):533–549, 1986.

[Gmytrasiewicz and Doshi, 2005] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *J. Artif. Intell. Res.*, 24:49–79, 2005.

[Goldman and Zilberstein, 2004] Claudia V. Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *J. Artif. Intell. Res.*, 22:143–174, 2004.

[Hatzack and Nebel, 2013] Wolfgang Hatzack and Bernhard Nebel. Solving the operational traffic control problem. In A. Cesta, editor, *Proceedings of the 6th European Conference on Planning (ECP-01)*, Toledo, Spain, 2013. AAAI Press.

[Kornhauser *et al.*, 1984] Daniel Kornhauser, Gary L. Miller, and Paul G. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *25th Annual Symposium on Foundations of Computer Science (FOCS-84)*, pages 241–250, 1984.

[Lawrence and Bulitko, 2013] Ramon Lawrence and Vadim Bulitko. Database-driven real-time heuristic search in video-game pathfinding. *IEEE Trans. Comput. Intellig. and AI in Games*, 5(3):227–241, 2013.

[Löwe *et al.*, 2011] Benedikt Löwe, Eric Pacuit, and Andreas Witzel. DEL planning and some tractable cases. In *Logic, Rationality, and Interaction - Third International Workshop (LORI-11)*, pages 179–192, 2011.

[Luna and Bekris, 2011] Ryan Luna and Kostas E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 294–300, 2011.

[Ma and Koenig, 2017] Hang Ma and Sven Koenig. AI buzzwords explained: multi-agent path finding (MAPF). *AI Matters*, 3(3):15–19, 2017.

[Muise *et al.*, 2015] Christian J. Muise, Vaishak Belle, Paolo Felli, Sheila A. McIlraith, Tim Miller, Adrian R. Pearce, and Liz Sonenberg. Planning over multi-agent epistemic states: A classical planning approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, pages 3327–3334, 2015.

[Ratner and Warmuth, 1986] Daniel Ratner and Manfred K. Warmuth. Finding a shortest solution for the N × N extension of the 15-puzzle is intractable. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, pages 168–172, 1986.

[Surynek, 2010] Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI-10)*, 2010.

[van den Berg *et al.*, 2008] Jur P. van den Berg, Ming C. Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation (ICRA-08)*, pages 1928–1935, 2008.

[Wang and Botea, 2011] Ko-Hsin Cindy Wang and Adi Botea. MAPP: A scalable multi-agent path planning algorithm with tractability and completeness guarantees. *J. Artif. Intell. Res.*, 42:55–90, 2011.

[Wurman *et al.*, 2008] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1):9–20, 2008.

[Yu and LaValle, 2013] Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the Twenty-Seventh Conference on Artificial Intelligence (AAAI-13)*, 2013.