Verification and Automated Construction of Social Laws for MAPF (Student Abstract)

Jakub Mestek
Charles University
Faculty of Mathematics and Physics
mestek@ktiml.mff.cuni.cz

Introduction

The task of Multi-Agent Pathfinding (MAPF; Stern et al. 2019) is to navigate agents (robots, cars, etc.) to their destinations while ensuring they do not collide with each other. We consider a discrete grid-like environment and discrete time. In each timestep, every agent can either move to a neighboring cell or stay at its current location.

The problem of computing an optimal solution (in terms of minimizing the time until agents reach their goals) is NP-complete (Yu and LaValle 2013), therefore fast (although not optimal) algorithms are desired.

The common centralized approach – the multi-agent plan is computed in advance by a central solver – works well in applications such as automatic warehouses. For some other applications, the decentralized approach – every agent makes decisions on its own – might be better suited (e.g., large environments with limited long-range communication, new agents dynamically appearing and disappearing).

This work deals with a novel decentralized approach built on the concept of social laws (Shoham and Tennenholtz 1995) – a common ordered set of rules that restrict agent actions based on the current state (situation of the agent) to guarantee a succesful co-existence of agents in multi-agent environments. Our recent work (Slezák, Mestek, and Barták 2025) applies this idea to MAPF, proposing a decentralized framework, where predefined laws (ordered rules) are used to avoid collisions. This work defines the key properties of such laws and proposes verification procedures, which are necessary for automatic construction of the laws.

Background on Social Laws for MAPF

In this decentralized framework, a social law is an ordered set of rules that, in applicable situations, prescribe actions to the agents. At each timestep, the agent executes the action prescribed by the first applicable rule. If no rule is applicable, the agent follows the shortest path to its goal. Figure 1 illustrates this process.

Each rule is a pair $\langle X, Y \rangle$ where Y is the prescribed action and X is a precondition specifying the applicable situations. Formally, X is a conjunction of conditions of the form "there is (or is not) an agent (or obstacle) at location L in the agent's neighborhood." The locations are specified relative to the position and orientation of the agent (orientation is defined by the next step on the shortest path). A 5×5 neighborhood centered at the agent is considered.

To foster deadlock resolution, the framework also supports stochastic laws, where the Y part of each rule is a probability distribution over prescribed actions, from which the actual action is sampled.

Related Work and Key Distinctions Many rule-based solvers have been proposed, such as Push and Swap (Luna and Bekris 2011) or PIBT (Okumura et al. 2022). Although quite fast, those solvers are centralized, hence requiring communication between the agents and the central solver.

The prior decentralized (i.e., mapping the current situation of the agent to action) MAPF approaches exist, but are learning-based, using neural networks (Sartoretti et al. 2019; Skrynnik et al. 2024), where the policies are hard to interpret and verify. In contrast, our rule-based framework is explicitly designed to be analyzable and formally verifiable.

Motivation for Automatic Construction Existing social laws (Slezák, Mestek, and Barták 2025) were manually designed and although verified and empirically promising, they

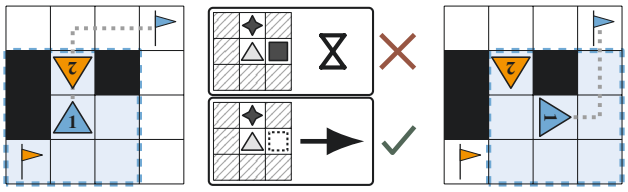

Figure 1: Illustrative toy example of the social laws framework, using only a 3×3 neighborhood. A conflicting situation (left) is resolved by a social law (center). The first rule (*if there is an agent ahead, regardless of its orientation, and an obstacle on the right, then wait*) does not apply to agent 1. The second rule (*if there is an agent ahead and no obstacle on the right, then move to the right*) applies, so agent 1 moves to the right (right), instead of following the shortest path to its goal (gray dotted line). The shortest path is then replanned, which also determines the agent's updated orientation. Regarding agent 2, the first rule applies, so agent 2 waits.

leave significant room for improvement (better success rate, lower time to reach target locations, and possibly utilizing a larger neighborhood). To achieve that, we need means of automatic construction of the laws.

Required Properties and Verification

Improving the laws and developing an algorithm for their automatic construction requires formal verification of candidate laws. Simulation-based evaluation is insufficient, as it provides no guarantees over all possible interactions.

The key requirement is safety, i.e., collision-free execution. A collision might be caused either by an agent following the shortest path (i.e., no rule was applied in the previous timestep) or by an agent that executed an action prescribed by the applied rule, motivating Coverage and Correctness.

Coverage requires that in every dangerous situation (i.e., the cell in front of the agent is occupied or another agent is heading there), a rule be applicable to at least one of the involved agents. Thus, Coverage prevents collisions that would arise if both agents simply followed their shortest paths.

Note that Coverage can be ensured trivially by appending, as the last rule, a catch-all rule with empty preconditions and the action "move forward". Since the agent's orientation is defined by the shortest path, this action exactly corresponds to following that path.

Correctness requires that an application of a law does not cause a collision (with other agents or obstacles). At the level of individual rules, Local Correctness requires that every prescribed action leads to a cell that is guaranteed (by the precondition) to be free of obstacles and other agents[1]; this can be checked by inspecting each rule independently.

However, interactions between locally correct rules can still lead to collisions, e.g., two agents prescribed to enter the same free cell. The social law must therefore also satisfy Concurrent Correctness: simultaneously applied rules must not send agents to the same cell.

A naive verification by enumerating all local situations in which two agents are located next to a free cell, and checking that conflicting actions are not prescribed, is infeasible due to the exponential number of cases (more than 3^{24} for 5×5 local neighborhood).

Instead, we propose to verify Concurrent Correctness by considering all potentially dangerous relative positions of two agents (about 48 such pairs, including orientation) and identifying pairs of rules that prescribe conflicting actions. For each such pair, we attempt to construct a local situation in which both rules would be simultaneously applied, i.e., preconditions are met and no higher-priority rules apply. This construction can be formulated using set operations over rule preconditions and is computable in polynomial time (w.r.t. the size of the neighborhood and number of rules). If such a situation exists, the social laws do not satisfy Concurrent Correctness.

[1]The ego-agent does not see the complete neighborhood of the other agent, hence we cannot guarantee that the other agent will leave.

Ideas on Automatic Construction

Correctness and Coverage guarantee safety (i.e., absence of collisions), but do not address efficiency – how quickly agents reach their targets, or whether they even avoid deadlocks. A successful automatic law-construction procedure must therefore produce laws that are not only safe, but also efficient. Unlike in the case of safety, simulation appears to be a suitable tool for evaluating efficiency.

One possible approach to automatic construction is to first construct rules that are efficient but potentially unsafe, e.g., by imitation learning from optimal MAPF solutions, and then systematically repair them until Coverage and Correctness are satisfied. The verification procedures outlined above could be adapted to guide this repair process.

Alternatively, one can formulate the construction as a constraint satisfaction problem that enforces Correctness and Coverage as hard constraints, thereby ensuring safety by design. Efficiency can then be encouraged using soft constraints, for example by preferring actions that follow the shortest path to the target on a set of representative instances.

These directions suggest that a combination of learning, constraint-based synthesis, and formal verification is a promising path toward automated design of social laws.

Acknowledgments

Research is supported by the Grant Agency of Charles University (project GAUK No. 36124) and by the Czech-Israeli Cooperative Scientific Research Project LUAIZ24104.

References

Luna, R.; and Bekris, K. E. 2011. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Proc. IJCAI 2011*, 2944–2949.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310: 103752.

Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.; Koenig, S.; and Choset, H. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters*, PP: 1–1.

Shoham, Y.; and Tennenholtz, M. 1995. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1): 231–252.

Skrynnik, A.; Andreychuk, A.; Nesterova, M.; Yakovlev, K.; and Panov, A. 2024. Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning. In *Proc. AAAI*, volume 38, 17541–17549.

Slezák, J.; Mestek, J.; and Barták, R. 2025. Social Laws for Multi-Agent Pathfinding. In *Proc. ICAART 2025*, 463–470. INSTICC, SciTePress.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Barták, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Symposium on Combinatorial Search (SoCS)*, 151–158.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. *IEEE Transactions on Robotics*, 29(6): 1444–1457.