# Social Laws for Multi-Agent Pathfinding

Jan Slezák, Jakub Mestek[a] and Roman Barták[b]
*Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic*
*slezak.jan33@gmail.com, {mestek, bartak}@ktiml.mff.cuni.cz*

Keywords: Path-Finding, Social Laws, Multi-Agent Environment.

Abstract: Multi-agent pathfinding (MAPF) is the problem of finding collision-free plans for navigating a set of agents from their starting positions to their destinations. Frequently, rigid plans for all agents are centrally searched and perfect execution of actions is assumed, which raises issues with the uncertainty and the dynamicity of the environment. Social laws are a well-established paradigm in multi-agent systems to coordinate agents that avoid extensive negotiation. In this work, we present a decentralized online algorithm to solve the MAPF problem without the need for communication between agents. This approach uses social laws inspired by traffic rules.

## 1 INTRODUCTION

Multi-agent pathfinding (MAPF) deals with navigating a set of agents in a shared environment. The task of MAPF is to find plans for agents that take them from their starting locations to their destinations while ensuring that the agents do not collide with each other (Stern et al., 2019). The environment is represented by a graph $G$ (vertices model locations and edges connections between them), and time is assumed to be discrete. In each time step $t$, every agent is located in a vertex of $G$, and between two consecutive time steps, every agent performs one of the possible actions, either *wait* at the current vertex or *move* to a neighbor vertex. Various forms of collisions have been defined (Stern et al., 2019), but in general, two agents should not be simultaneously at the same location (vertex or edge). MAPF has numerous applications in areas such as automated warehouses, crossroads control, and navigation of non-player characters in video games.

When executing the plans, the perfect synchronization of agents is assumed, which is unrealistic in environments with some uncertainty and dynamicity. Robust plans are generated to handle, for example, delays during execution (Atzmon et al., 2020) or robust execution strategies are used (Hönig et al., 2016). While robust plans are less flexible and assume the worst-case scenario, robust execution assumes explicit exchange of information between agents.

[a] https://orcid.org/0009-0003-8765-5092
[b] https://orcid.org/0000-0002-6717-8175

This paper presents a decentralized approach to solving MAPF without explicit agent communication. We use so-called social laws that prescribe or deny specific actions based on the agent's current local neighborhood, which corresponds to the agent's sensor input. In particular, we propose a formal framework to describe social rules for MAPF problems and define a set of social laws for MAPF that prevent the execution of any action that might cause a collision. If no such law is applicable in the given situation (i.e., no possible collision), the agent follows the shortest path to its destination. Both deterministic and stochastic rules are proposed, where the stochastic rules are intended to prevent deadlocks. The algorithm works online; at each time step, each agent decides on its next action. This online nature also makes the method applicable to non-deterministic settings where agents can experience delays.

The major contribution of the paper is bringing the concept of social laws to multi-agent path finding, which has not been done before. In particular, the formal framework to specify social laws for MAPF is suggested together with examples of two social laws, one deterministic and one stochastic. The proposed laws are empirically compared.

## 2 PROBLEM DEFINITION

A *Multi-Agent Pathfinding* problem with $k$ agents is a quadruple $(G, k, s, g)$ where $G = (V, E)$ is an undirected graph, $\{1, \dots, k\}$ the set of agents, $s$ and $g$ map-

Slezák, J., Mestek, J. and Barták, R.
Social Laws for Multi-Agent Pathfinding.
DOI: 10.5220/0013189200003890
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 1*, pages 463-470
ISBN: 978-989-758-737-5; ISSN: 2184-433X
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

pings $\{1,\dots,k\} \mapsto V$ that assign to each agent $i$ its starting vertex $s(i)$ and its destination (goal) $g(i)$, respectively.

Time is assumed to be discrete and, at each time step, each agent is located on one of the vertices $V$. Between two consecutive timesteps, each agent performs exactly one action. An *action* is modeled as a function $a\colon V \mapsto V$ where $a(v) = v'$ means that if an agent performs action $a$ in the vertex $v$ at time step $t$, in the next time step $t+1$ the agent will be in the vertex $v'$. There are two types of actions: the *wait* action means that the agent stays at its current vertex ($v' = v$), and the *move* action means that the agent moves to an adjacent vertex ($(v, v') \in E$) (Stern et al., 2019).

For agent $i$ and a sequence of actions $\pi = (a_1, a_2, \dots, a_n)$, $\pi_i[t]$ denotes the location of the agent after performing the first $t$ actions of $\pi$, starting at its starting vertex $s(i)$. As we assume unit durations of all actions, $\pi_i[t]$ represents the location of agent $i$ at the time step $t$. A sequence $\pi$ is a *single-agent plan* for agent $i$ iff $\pi_i[|\pi|] = g(i)$, that is, the execution of the actions starting at $s(i)$ results in being at $g(i)$.

A *solution* is a set of $k$ single agent plans, one for each agent. The task of MAPF is to find a *valid* solution, that is, a solution that does not contain any conflict (collisions during execution). In this paper, we recognize three types of conflict:

- A *vertex conflict* between two agents $i$ and $j$ occurs if both agents occupy the same vertex at the same time step. Formally, there exists $t$ such that $\pi_i[t] = \pi_j[t]$.
- An *edge conflict* between two agents $i$ and $j$ occurs if both agents move along the same edge at the same time. Formally, there exists $t$ such that $\pi_i[t] = \pi_j[t] \land \pi_i[t+1] = \pi_j[t+1]$. Note that the existence of an edge conflict implies the existence of a vertex conflict; therefore, it is sufficient to check only for vertex conflicts.
- A *swapping conflict* between two agents $i$ and $j$ occurs if the agents move at the same time along the same edge in opposite directions. Formally, there exists $t$ such that $\pi_i[t] = \pi_j[t+1] \land \pi_i[t+1] = \pi_j[t]$.

**Target Behavior.** We use the *disappear at target* assumption – agent disappears upon reaching its destination. This means that the agent does not block other agents after it reaches its destination.

**Solution Cost.** To measure the cost ("quality") of the solutions, we use the *Sum of Cost (SoC)* metric. SoC of a given solution is defined as the sum of the lengths of the single-agent plans.

## 3 RELATED WORK

MAPF has been extensively studied and many complete optimal algorithms exist, e.g., Conflict-Based Search (Sharon et al., 2015) and its improvements are very popular. Such algorithms are inherently offline and centralized, which limits their applicability in dynamic environments (e.g., where agents can get delayed) and scalability in number of agents. Decentralized and on-line algorithms might be an answer to those issues.

Priority Inheritance with Backtracking (PIBT) (Okumura et al., 2019) is a complete, optionally decentralized, algorithm based on prioritized planning; decentralized implementation requires local communication. In prioritized planning, priorities are assigned to agents and single-agent plans are computed one by one (in the order given by agents' priorities) while avoiding collisions with already computed plans. Note that priorities are usually artificially chosen, just to specify the ordering of the agents. (Ma et al., 2019) showed that static (a priory fixed) priorities are not sufficient to achieve completeness. To resolve deadlocks and thus achieve completeness, PIBT uses priority inheritance and backtracking which can be seen as a form of search in the space of mappings of agents to priorities and is being done at every time step.

A different approach to the decentralized and online solution of MAPF is reinforcement learning. The PRIMAL algorithm (Sartoretti et al., 2019) learns a policy that outputs an action for the agent based on its current situation represented by the agent's neighborhood ($10 \times 10$) and distance and direction to its goal. In the Follower algorithm (Skrynnik et al., 2024) each agent individually computes the shortest path to its destination, and then the learned policy is used to follow the path, ensuring collision avoidance by making the necessary detours.

Social laws were introduced as a general concept that guarantees successful co-existence of agents in multi-agent environments by denying some actions (Shoham and Tennenholtz, 1995). More formally, a social law was defined as an ordered set of constraints $(a_i, \varphi_i)$ where $(a_i, \varphi_i)$ means that action $a_i$ is not allowed in states that satisfy condition $\varphi_i$. The idea of using social laws was later extended to multi-agent planning setting MA-STRIPS where social laws modify the MA-STRIPS instances – for example, add or remove actions, add or remove preconditions or effects, etc. (Karpas et al., 2017).

## 4 SOCIAL LAWS FOR MAPF

We introduce social laws for MAPF as rules that prescribe actions to execute based on the current situation in the local neighborhood of the agent. Contrary to the original definition, we use the laws in an *online* approach: each agent follows the shortest path to its goal, and social laws are used only to resolve potential conflicts with other agents. However, we retain the original idea of social laws as a method of achieving coordination without the need to negotiate (and communicate explicitly).

### 4.1 Agent Neighborhood

Though graphs that define the environment of agents could be arbitrary, the most common way is to use grid-like graphs, where agents can move up, down, left or right. We use this concept in our definition of agent's neighborhood as it allows the definition of social laws and forbidden moves independently of the particular vertex of the agent.

To describe the situation in a local neighborhood, we index the vertices in the following way: The vertex where the agent is currently located is marked by 0. The vertex to which the next planned action leads (where the agent is headed) is marked by 1. Then, we mark the rest of the 8-neighborhood of 0 by 2–8, clockwise. Similarly, the next vertex in the direction of 1 is marked by 9, and the rest of the $5 \times 5$-neighborhood by 10–24. The labeling is shown in Figure 1. This allows us to describe a neighborhood (visible area) of different sizes.

| | | | | |
|---|---|---|---|---|
| 23 | 24 | 9 | 10 | 11 |
| 22 | 8 | 1 | 2 | 12 |
| 21 | 7 | 0 | 3 | 13 |
| 20 | 6 | 5 | 4 | 14 |
| 19 | 18 | 17 | 16 | 15 |

Figure 1: Labeling of local neighborhood of an agent.

The situation in the neighborhood is then described using the prepositions $F = \{0,\ldots,24\} \times S$ where $S = \{A,N,O,P\}$ is a status of the vertex:

$A$ – occupied by an agent

$N$ – not occupied by an agent (i.e. obstacle or free)

$O$ – obstacle

$P$ – passable (i.e. not obstacle)

A set of actions $A = \{S,F,R,L,B\}$ is again relative to location and the next planned action of the agent:

$S$ – stay at 0

$F$ – move forward in the planned direction, i.e., to 1

$R$ – turn right, i.e., move to 3

$L$ – turn left, i.e., move to 7

$B$ – move back, i.e., to 5

A *law* $P$ is a pair $\langle X,Y \rangle$ where $X \subseteq F$ are preconditions of the law, and $Y$ is a probabilistic distribution of actions $A$ (we allow laws with stochastic effects; such laws are called stochastic laws). A law is *applicable* in the current situation of the agent if all its preconditions are satisfied.

*Social laws* are a totally ordered set of such laws.

### 4.2 Usage of the Laws

Now, we describe how to use social laws to control the agents. We consider a decentralized setting, meaning that every agent is controlled by itself. We assume that each agent knows the map and its current position (so it can compute the shortest path to its destination), and also knows the positions of agents within its local neighborhood (sensor input).

In the beginning, the agent computes a shortest path from its starting location to its destination and follows the path. At each time step, the agent checks all social laws in a given order. If there is an applicable law, the first such law is applied, that is, the agent performs the action suggested by the law. In the case of stochastic laws, the action is sampled randomly from the distribution. If the new location of the agent is not on the original shortest path to the destination, the agent computes a new shortest path – from the current location to the agent's destination. During shortest path computation, other agents are ignored (not considered as obstacles), and hence possible conflicts with other agents are ignored.

If none of the laws are applicable, the agent continues to the next vertex on the shortest path.

Note that each agent acts independently based solely on the situation (positions of other agents and obstacles) in its neighborhood. The agent does not consider other agent's intentions or possible actions. Hence, the computational complexity of determining the next action for an agent is constant. In particular, it does not depend on the number of agents in the environment.

### 4.3 Deterministic Laws

We now describe a basic system of social laws based on the principle of "priority to the right." In the defini-

```
 1: (A1) (N3) (P3) (N4) (N13) > (R100)
 2: (A1) (N5) (P5) (N6) (N17) > (B100)
 3: (A1) > (S100)
 4: (A2) (N3) (P3) (N4) (N13) > (R100)
 5: (A2) (N5) (P5) (N6) (N17) > (B100)
 6: (A2) > (S100)
 7: (A9) (N3) (P3) (N4) (N13) > (R100)
 8: (A9) (N5) (P5) (N6) (N17) > (B100)
 9: (A9) > (S100)
10: (A10) (N3) (P3) (N4) (N13) > (R100)
11: (A12) (N3) (P3) (N4) (N13) > (R100)
12: (A3) (N5) (P5) (N6) (N17) > (B100)
13: (A11) (N3) (P3) (N4) (N13) > (R100)
14: (A13) (N5) (P5) (N6) (N17) > (B100)
```

Figure 2: Deterministic Social Laws (action selection has probability 100%) – a complete system.

tions of the laws, we use the aforementioned relative labeling of the vertices (Figure 1).

**Do not Enter Occupied Vertex Law (NO).** The agent at 0 is not allowed to enter the planned vertex 1, if the vertex 1 is already occupied by another agent (laws 1-3).

**Priority to the Right Law (PR).** The agent at 0 is not allowed to enter the planned vertex 1 if there is another agent at 2 (laws 4-6). Furthermore, to prevent a collision at 1 with an agent incoming from the opposite direction (9), we also do not allow the agent to enter 1 if there is an agent at 9 and no agent at 8.

We want to ensure that when the agent is not allowed to enter its planned vertex (because of another agent), it will try to "get around" the other agent. Therefore, the agent first tries to move to the right (to 3). If not possible[1] (due to another agent or obstacle), the agent tries to move back (to 5). Only if none of the moves is possible, the agent stays at 0.

The complete system of above rules is listed in Figure 2. We will refer to this system as Deterministic Social Laws (DSL).

### 4.3.1 Properties

DSL ensures that no collisions occur during execution. Proof: The swapping conflict is denied by the NO rule; the agent is not allowed to move towards the vertex, which is occupied by another agent. Vertex conflict is also denied, as NO does not allow an agent to enter an already occupied vertex, and PR ensures that no two agents move to the same vertex at the same time step.

[1]Vacancy of the new location is not sufficient. The possible violation of PR or NO by the new action is checked as well.

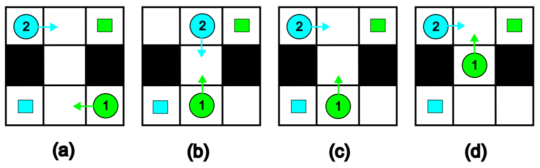

Figure 3: Deadlock situation and its possible resolution using stochastic rules. The current positions of agents are depicted by circles, their destinations by squares. Arrows denote agents' planned actions.

DSL also ensures robustness against delays, as the occupancy of the next vertex is checked in real-time.

On the other hand, DSL is prone to ending in a deadlock, due to the symmetrical behavior of all involved agents. An example of such a situation is shown in Figures 3a, 3b. Using DSL, in situation (b), the PR law is applicable to both agents, so they both move right (relatively to their desired directions depicted by arrows). Therefore, at the next time step, agents are again in situation (a).

As a solution to resolve such deadlocks, we propose another system that uses stochastic laws. Stochasticity breaks the symmetrical behavior of agents.

## 4.4 Stochastic Laws

Stochastic laws are modifications of the deterministic laws. The precondition part remains the same, while the action part is modified such that instead of forcing the agent to always move right (or back, respectively), we set the agent to stay, with 50% probability. This specific value maximizes the probability that two agents choose different actions (in symmetric situations such as the one in Figure 3b).

To foster deadlock avoidance, we introduce an extended version of the priority to the right law.

**Extended Priority to the Right Rule (EPR).** In addition to situations described in standard Priority to the right: An agent in 0 is allowed to enter 1 when there is an agent in 9 if there is an agent in 2, an obstacle in 3 and no obstacle at 24. The agent in 0 is also allowed to enter 1 when there are agents in all neighbors of 1 (i.e., 9, 2, 8), an obstacle in 3 and no obstacles in 3, 7, 24.

Furthermore, in addition to trying to move right and back, the agent also tries to move left.

A complete system of Stochastic Social Laws (SSL) is shown in Figure 4. It consists of stochastic versions of the EPR and NO laws.

```
 1: (A1) (N3) (P3) (N4) (N13) > (R50) (S50)
 2: (A1) (N5) (P5) (N6) (N17) > (B50) (S50)
 3: (A1) (N7) (P7) (N8) (N21) > (L50) (S50)
 4: (A1) (N3) (P3) (A2) (A13) (N4) > (R50) (S50)
 5: (A1) (N3) (P3) (O5) (A13) (P12) (N4) > (R50) (S50)
 6: (A1) (N3) (P3) (A2) (P1) (O5) (A13) (P12) (A4) (P14) > (R50) (S50)
 7: (A1) (N5) (P5) (A4) (A17) (N6) > (B50) (S50)
 8: (A1) (N5) (P5) (O7) (A17) (P16) (N6) > (B50) (S50)
 9: (A1) (N5) (P5) (A4) (P3) (O7) (A17) (P16) (A6) (P18) > (B50) (S50)
10: (A1) > (S100)
11: (A2) (N3) (P3) (N4) (N13) > (R50) (S50)
12: (A2) (N5) (P5) (N6) (N17) > (B50) (S50)
13: (A2) (N7) (P7) (N8) (N21) > (L50) (S50)
14: (A2) (N1) (P1) (A8) (P7) (O3) (A9) (P24) (A2) (P10) > (F50) (S50)
15: (A2) (N3) (P3) (A2) (A13) (N4) > (R50) (S50)
16: (A2) (N3) (P3) (O5) (A13) (P12) (N4) > (R50) (S50)
17: (A2) (N3) (P3) (A2) (P1) (O5) (A13) (P12) (A4) (P14) > (R50) (S50)
18: (A2) (N5) (P5) (A4) (A17) (N6) > (B50) (S50)
19: (A2) (N5) (P5) (O7) (A17) (P16) (N6) > (B50) (S50)
20: (A2) (N5) (P5) (A4) (P3) (O7) (A17) (P16) (A6) (P18) > (B50) (S50)
21: (A2) > (S100)
22: (A9) (N3) (P3) (N4) (N13) > (R50) (S50)
23: (A9) (N5) (P5) (N6) (N17) > (B50) (S50)
24: (A9) (N7) (P7) (N8) (N21) > (L50) (S50)
25: (A9) (N1) (P1) (A8) (A9) (N2) > (F50) (S50)
26: (A9) (N1) (P1) (O3) (A9) (P24) (N2) > (F50) (S50)
27: (A9) (N3) (P3) (A2) (A13) (N4) > (R50) (S50)
28: (A9) (N3) (P3) (O5) (A13) (P12) (N4) > (R50) (S50)
29: (A9) (N3) (P3) (A2) (P1) (O5) (A13) (P12) (A4) (P14) > (R50) (S50)
30: (A9) (N5) (P5) (A4) (A17) (N6) > (B50) (S50)
31: (A9) (N5) (P5) (O7) (A17) (P16) (N6) > (B50) (S50)
32: (A9) (N5) (P5) (A4) (P3) (O7) (A17) (P16) (A6) (P18) > (B50) (S50)
33: (A9) > (S100)
34: (A8) (O7) (O2) (O9) (P3) (N3) (N4) (N13) > (R50) (S50)
35: (A8) (O7) (O2) (O9) (P5) (N5) (N6) (N17) > (B50) (S50)
36: (A10) (N3) (P3) (N4) (N13) (P2) > (R50) (S50)
37: (A12) (N3) (P3) (N4) (N13) (P2) > (R50) (S50)
38: (A3) (N5) (P5) (N6) (N17) > (B50) (S50)
```

Figure 4: Stochastic Social Laws – a complete system.

#### 4.4.1 Properties

Similarly to PR, EPR does not allow two agents to move simultaneously to the same vertex. Therefore, SSL also ensures that no collisions occur during execution, as well as robustness against delays.

SSL is much less prone to deadlocks. For example, the situation in Figure 3 is solvable using SSL because with probability 50%, exactly one of the agents stays and the other moves right, leading to a situation (c) (or symmetric where the blue agent stayed and the green agent moved). The agents can then proceed to situation (d) and continue to their destinations. Note that, due to stochasticity, with a probability 50% either both or none of the agents move. That would lead to repeating one of the situations (a) or (b), thus just needing another attempt to solve the situation.

However, there are still some instances that cannot be solved using SSL (or DSL); one such situation is shown in Figure 5. To solve such instances by stochastic rules, adding (unique) priorities to the agents or involving some form of communication is the way to handle the situation.

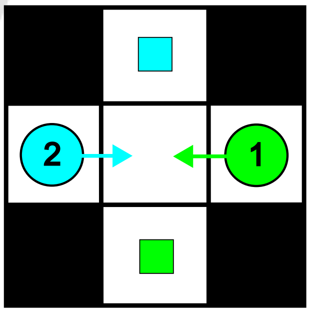

Figure 5: Situation not solvable by social laws.

## 5 EMPIRICAL EVALUATION

We evaluated our proposed method on maps and scenarios from the standard MAPF benchmark (Stern et al., 2019). We compare the results of our method with optimal solutions for given instances from MAPF Progress Tracker (Shen et al., 2023) and with the decentralized ML-based algorithm Follower (Skrynnik et al., 2024).

## 5.1 Comparison of DSL and SSL

Firstly, we experimentally compare the performance of the two proposed social law systems, namely DSL and SSL, on the empty grid of size $32 \times 32$ with various numbers of agents. Success rates are shown in Table 1. The results clearly indicate that the basic system of deterministic laws (DSL) is prone to ending in a deadlock. In contrast, using stochastic laws (SSL), we were able to solve all instances.

Comparison of the costs (Table 2) shows that SSL are better even in terms of the costs of the solutions. Therefore, in next experiments, we will use the SSL law only.

Table 1: DSL and SSL success rates on map *empty-32-32*.

| Agents | DSL Success Rate | SSL Success Rate |
|---|---|---|
| 10 | 1,00 | 1,00 |
| 20 | 0,96 | 1,00 |
| 30 | 0,92 | 1,00 |
| 40 | 0,84 | 1,00 |
| 50 | 0,84 | 1,00 |

Table 2: DSL and SSL plan costs on map *empty-32-32*.

| Agents | DSL Average SoC | SSL Average Soc |
|---|---|---|
| 10 | 227,20 | 225,40 |
| 20 | 501,38 | 490,04 |
| 30 | 836,61 | 806,96 |
| 40 | 1204,00 | 1155,10 |
| 50 | 1670,57 | 1580,86 |

## 5.2 Performance

We evaluated SSL on several maps and numbers of agents; for every such pair, we present average values over the 25 random scenarios from MAPF Benchmark. We report the SoC of SSL, the ratio of that SoC compared to the SoC of the optimal solution (from (Shen et al., 2023)), and the number of applied social laws and the number of replannings. Note that replanning occurs when an action different from stay or move forward is enforced by a social rule. In such a case, a new path for a single agent is sought.

Tables 3 and 4 show the performance of SSL on the empty grid $32 \times 32$ and the grid with 20% random generated obstacles, respectively. On the map with obstacles, the average number of applications of a social rule is significantly higher which means more interactions between agents. This fact is expected and is probably the main explanation for the increase in the ratio to optimal costs.

The other two sets of experiments were performed on maps inspired by a warehouse layout. Table 5 shows the results on the warehouse map with corridors of width 1. Although SSL is able to solve such instances, the costs of the solutions are several times worse than the optimum. In contrast, on a similar map but with corridors of width 2 (so that agents can pass each other), SSL is quite competitive, even for many agents (900). The results of such maps are shown in Table 6.

The Follower algorithm achieved almost optimal performance on most of the maps (except the warehouse with narrow corridors), thus being much better than our approach. However, the Follower uses the local neighborhood of size $10 \times 10$ which is much larger than ours ($5 \times 5$). This opens an interesting option for future research, the construction of social laws that use a larger neighborhood. Furthermore, in our approach, the laws are formally defined and verifiable.

## 6 CONCLUSION

In this paper, we introduce the idea of solving the MAPF using social laws. This approach is decentralized; we assume that each agent knows only the map (to compute the shortest path to its goal) and positions of agents (and obstacles) in its local neighborhood. We do not assume any communication between agents. We proposed a framework to describe social laws in this setting formally.

In the framework, we designed two systems of laws, Deterministic Social Laws (DSL) and Stochastic Social Laws (SSL). Due to the symmetric behavior of agents, the Deterministic Social Laws can easily end up in a deadlock, which was shown both theoretically and experimentally. To combat the symmetricity, we designed Stochastic Social Laws – stochasticity makes it possible for two agents in symmetric situations to perform different actions.

The Stochastic Social Laws were experimentally evaluated on four maps (grid with and without obstacles, warehouse with narrow and wide corridors) from the standard MAPF Benchmark. We found that SSL solved all the tested instances. Regarding the cost of the solutions, we compare the SSL solutions to the optimal solutions. With an increasing number of agents, the ratio of costs increases significantly. We found

Table 3: Performance of SSL on *empty-32-32*.

| Agents | Average SoC | Rating | Average Replans | Average Social Laws | Follower SoC |
|---|---|---|---|---|---|
| 10 | 225,40 | 1,06 | 5,68 | 11,36 | 212,92 |
| 20 | 490,84 | 1,14 | 26,72 | 53,48 | 435,84 |
| 30 | 809,96 | 1,25 | 72,68 | 145,44 | 663,16 |
| 40 | 1150,84 | 1,35 | 131,42 | 263,20 | 881,72 |
| 50 | 1563,28 | 1,45 | 214,78 | 430,56 | 1119,60 |
| 100 | 4576,52 | 2,12 | 1050,12 | 2119,96 | 2356,80 |
| 150 | 10832,28 | 3,33 | 3146,86 | 6513,24 | 3739,84 |
| 200 | 20312,32 | 4,67 | 6376,32 | 13693,24 | 5274,64 |
| 250 | 34772,24 | 6,31 | 11286,74 | 25209,52 | 7025,24 |
| 300 | 58683,04 | 8,62 | 18980,60 | 45131,76 | 9048,12 |
| 350 | 98451,84 | 11,78 | 30708,02 | 79629,48 | 11444,28 |
| 400 | 157456,72 | 15,49 | 46115,18 | 132260,80 | 14464,08 |
| 450 | 249707,72 | 19,83 | 67320,46 | 216599,68 | 18390,32 |
| 500 | 417161,20 | 27,11 | 98268,34 | 373848,56 | 23259,44 |

Table 4: Performance of SSL on *random-32-32-20*.

| Agents | Average SoC | Rating | Average Replans | Average Social Laws | Follower SoC |
|---|---|---|---|---|---|
| 10 | 255,20 | 1,13 | 11,62 | 23,48 | 230,68 |
| 20 | 570,40 | 1,27 | 46,52 | 94,96 | 465,68 |
| 30 | 930,20 | 1,37 | 98,30 | 201,56 | 713,96 |
| 40 | 1420,40 | 1,55 | 195,60 | 407,40 | 974,80 |
| 50 | 1914,24 | 1,68 | 300,72 | 629,84 | 1228,84 |
| 100 | 6280,44 | 2,70 | 1467,68 | 3220,48 | 2688,96 |
| 150 | 15293,24 | 4,15 | 4094,02 | 9614,96 | 4656,68 |
| 200 | 30665,00 | 5,86 | 8491,00 | 21399,56 | 7036,60 |
| 250 | 60069,28 | 8,41 | 16193,18 | 45476,52 | 10268,72 |
| 300 | 102401,64 | 10,16 | 25909,70 | 81746,28 | 14816,24 |
| 350 | 167805,00 | 12,36 | 38806,82 | 140010,88 | 21043,16 |
| 400 | 266375,96 | 9,08 | 56306,98 | 229531,80 | 29262,08 |

Table 5: Performance of SSL on *warehouse-10-20-10-2-1*.

| Agents | Average SoC | Rating | Average Replans | Average Social Laws | Follower SoC |
|---|---|---|---|---|---|
| 100 | 15374,04 | 1,77 | 2063,16 | 4739,56 | 9238,88 |
| 300 | 80707,00 | 3,28 | 15889,56 | 42504,00 | 27777,36 |
| 500 | 334662,00 | 7,56 | 70588,92 | 234236,72 | 50819,96 |
| 700 | 1132945,00 | 16,30 | 229907,68 | 887009,12 | 79063,20 |

Table 6: Performance of SSL on *warehouse-10-20-10-2-2*.

| Agents | Average SoC | Rating | Average Replans | Average Social Laws | Follower SoC |
|---|---|---|---|---|---|
| 100 | 9687,16 | 1,05 | 196,08 | 392,48 | 9378,24 |
| 300 | 30556,84 | 1,17 | 1812,96 | 3648,80 | 27218,48 |
| 500 | 60630,80 | 1,39 | 6879,74 | 14082,16 | 46193,84 |
| 700 | 105976,56 | 1,67 | 17222,98 | 36080,16 | 66994,08 |
| 900 | 171310,44 | 1,94 | 34574,56 | 74768,84 | 88691,64 |

that social rules work well on maps where agents are able to easily avoid each other (such as warehouses with wide corridors).

Our work opens several directions for future research. We limited our research only to laws inspired by the priority to the right rule that is used in real-world traffic rules. However, systems based on different rules might be more efficient or even needed if human agents are assumed.

We showed an example of an instance that is not solvable using social laws (in our current setting). Some extensions to the setting, such as adding priorities to the agent, would not only make such instances solvable, but it might also be useful for decreasing the cost of solutions produced by social laws.

## ACKNOWLEDGEMENTS

Research is supported by the project 23-05104S of the Czech Science Foundation. Jakub Mestek is supported by Grant Agency of Charles University (project GAUK No. 36124). Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth, and Sports of the Czech Republic.

## REFERENCES

Atzmon, D., Stern, R., Felner, A., Wagner, G., Barták, R., and Zhou, N. (2020). Robust multi-agent path finding and executing. *J. Artif. Intell. Res.*, 67:549–579.

Hönig, W., Kumar, T. K. S., Cohen, L., Ma, H., Xu, H., Ayanian, N., and Koenig, S. (2016). Multi-agent path finding with kinematic constraints. In Coles, A. J., Coles, A., Edelkamp, S., Magazzeni, D., and Sanner, S., editors, *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016*, pages 477–485. AAAI Press.

Karpas, E., Shleyfman, A., and Tennenholtz, M. (2017). Automated verification of social law robustness in STRIPS. In *Proceedings of the International Conference on Automated Planning and Scheduling, ICAPS 2017*, volume 27, pages 163–171.

Ma, H., Harabor, D., Stuckey, P. J., Li, J., and Koenig, S. (2019). Searching with consistent prioritization for Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7643–7650.

Okumura, K., Machida, M., Défago, X., and Tamura, Y. (2019). Priority Inheritance with Backtracking for Iterative Multi-agent Path Finding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 535–542. International Joint Conferences on Artificial Intelligence Organization.

Sartoretti, G., Kerr, J., Shi, Y., Wagner, G., Kumar, T. K. S., Koenig, S., and Choset, H. (2019). PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385.

Sharon, G., Stern, R., Felner, A., and Sturtevant, N. R. (2015). Conflict-based search for optimal Multi-agent Pathfinding. *Artificial Intelligence*, 219:40–66.

Shen, B., Chen, Z., Cheema, M. A., Harabor, D. D., and Stuckey, P. J. (2023). Tracking progress in multi-agent path finding. arXiv:2305.08446 [cs.AI].

Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1):231–252. Computational Research on Interaction and Agency, Part 2.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17541–17549.

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)*, pages 151–158.