2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Multi-Robot Navigation Among Movable Obstacles: Implicit Coordination to Deal with Conflicts and Deadlocks

Benoit Renault[a], Jacques Saraydaryan[b], David Brown[a] and Olivier Simonin[a]

***Abstract*— How to coordinate multiple robots moving in modifiable cluttered environments? In this paper, we introduce the multi-robot version of the NAMO problem (Navigation Among Movable Obstacles). In MR-NAMO, robots must not only plan for the possibility of displacing obstacles as needed to facilitate their navigation, but also solve conflicts that may arise when trying to simultaneously access a location or obstacle. After identifying all different types of conflicts, we define and compare variants of an implicit coordination strategy allowing the use of existing NAMO algorithms [1] in a Multi-Robot context. We also show how our previously introduced social occupation cost model [2] can improve the efficiency of multi-robot plans with better obstacle placement choices, and how it can be applied in a novel way to find relevant robot placement choices to solve deadlock situations.**

I. INTRODUCTION

To accomplish their tasks, humans do not hesitate to move objects out of their way. It has been established that providing robots with the same capability is a necessity, leading to the formulation of the Navigation Among Movable Obstacles (NAMO) problem: computing *a single robot*'s collision-free path from a start to a goal configuration, allowing the manipulation of movable obstacles and minimizing a displacement cost (e.g. travel distance, time, energy) [1].

In a previous state-of-the-art on NAMO, we have shown that existing approaches have never considered the presence of humans or other robots in the environment [3]. As such, current NAMO algorithms cannot trivially be applied to multi-robot problems, because they do not provide any coordination mechanisms that would preserve their fundamental no-collision guarantee. Beyond this main concern, we have found that the single-robot focus of existing NAMO algorithms led them to only minimize the robot's displacement cost. Thus, the robot would always move obstacles to the nearest place that opens its path, regardless of the impact on the overall environment structure: this can be observed in Fig. 1.a-c, where the blue robot leaves the obstacle in the entryway, forcing the pink robot to move it again (Fig. 1d).

To address this second issue, we previously introduced a Social Occupation Costmap [2], illustrated in Fig. 1e. We modified an existing NAMO planner [1] to find compromises between the robot's displacement cost and the social occupation cost, as to reduce space accessibility disturbance for humans/robots, regardless of their actual presence in the environment at plan-time.

[a]B. Renault, O. Simonin and D. Brown are with INSA Lyon, CITI Lab and INRIA Chroma Team benoit.renault@insa-lyon.fr, olivier.simonin@inria.fr, david.brown@inria.fr
[b]Jacques Saraydaryan is with CPE Lyon, CITI Lab and INRIA Chroma Team jacques.saraydaryan@cpe.fr

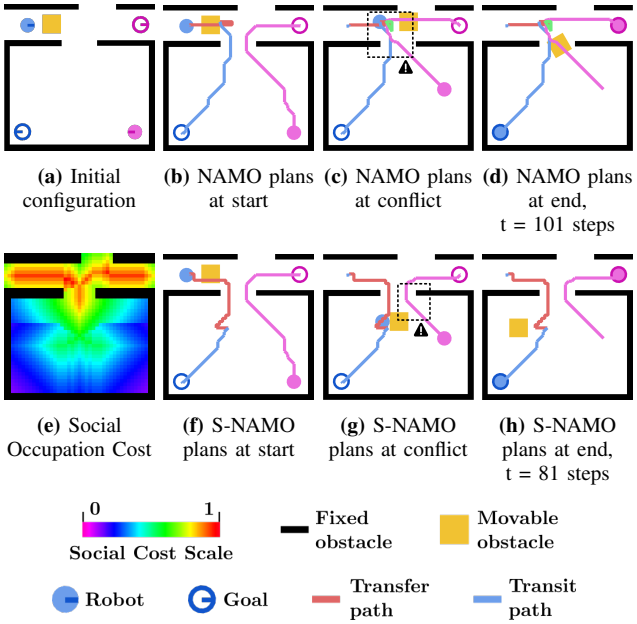


Fig. 1: Simple Multi-Robot scenario and resolution by our Implicit Coordination Strategy without (1st line) and with (2nd line) the Social-NAMO Model [2].

In this paper, we define the Multi-Robot NAMO problem (MR-NAMO) and present a generic implicit coordination strategy to use existing NAMO algorithms in multi-robot settings. To deal with deadlocks, we propose two local strategies to decide which robots move, and where, as to give way to the others. These strategies are respectively based on distance and social-cost optimization criteria. Then we conduct experiments, using our open source simulator extended for MR-NAMO, to evaluate these approaches. We show how a better space organization positively affects multi-robot NAMO performance criteria, such as the number of obstacle transfers and the total distance traveled. This is illustrated in Fig. 1.f-h, where the pink robot no longer moves the obstacle, reducing travel distance and execution time. We also analyze the efficiency and limits of the implicit coordination strategy, especially in relation to the number of robots in the environment.

This paper is organized as follows. Section II provides an overview of related work, with a short reminder of our Social Placement Cost Model. Section III formalizes a definition of the MR-NAMO problem and the inherent conflicts. Section IV introduces an implicit coordination algorithm with variations capable of detecting and avoiding conflicts while dealing with deadlocks. Finally, we compare and analyze the proposed approaches with simulated scenarios in Section V.

979-8-3503-7769-9/24/$31.00 ©2024 IEEE

We conclude and discuss future work in Section VI.

II. RELATED WORK

A. NAMO: a single-robot problem

NAMO algorithms usually solve the problem by interleaving an obstacle choice strategy with existing motion planners (e.g. A*, Dijkstra, RRT,...) to compute non-colliding navigation and manipulation paths [3]. Few offer completeness guarantees : only Stilman's Select-Connect algorithm [1] is resolution complete for the common class of L1, "Linear" problems where free space components can be connected independently by moving a single obstacle.

From the problem's inception [1], to the most recent papers [4], [5], NAMO has only ever been formulated as a single-robot problem. The only relative exception would be Mueggler et al.'s paper [6], where a drone hovers over, localizes and remotely controls a ground robot that executes a single-robot NAMO plan, but the algorithm could not trivially compute a collision-free path for multiple agents sharing a same space.

In a previous work, we introduced Social NAMO (S-NAMO) where the robot is additionally required to minimize a newly defined social occupation cost, representing the disturbance to the environment's accessibility affordance for humans, caused by transferred obstacles [2]. With only the binary occupancy grid of static obstacles (e.g. walls, heavy furniture), a social occupation costmap is derived, based on two heuristics: avoiding narrow areas, and the middle of space, yielding higher costs there (Fig. 1e). Our S-NAMO planner modifies Stilman's (2005) reference algorithm, so that it seeks a compromise cost between the weighted euclidean distance traversed by the robot and the social cost.

B. Multi-Robot related problems

While Multi-Robot NAMO has not been specifically addressed, related problems combining robot navigation and obstacle manipulation, do have multi-agent extensions.

One close problem is Multi-Robot Manipulation Planning and its simpler variant, the Multi-Agent Object-Pushing Problem [7]. Robots must move a single object together to a given position, focusing on cooperative manipulation. However, this is different from finding individual paths that move different objects.

Another interesting problem is Multi-Robot Rearrangement/Assembly Planning [8][9], a superset of the previous problem, where multiple movable objects are considered and must reach given target positions. Both problems however differ from the NAMO problem, since the final position is known and so reduce drastically the search space. Indeed, they do not compute the best final arrangement, which is where a large part of the NAMO problem's difficulty lies.

A third close problem of NAMO is Multi-Robot Combined Task and Motion Planning (TMP/TAMP). TMP is actually a superset of the NAMO problem [10]. However, current solutions to the overarching MR-TMP problem [11] do not actually confront movable obstacles the way NAMO does. For instance, Motes et al [11] only introduce navigation and handover tasks where handed objects do not use physical space.

Finally, Multi-Robot Coordination and Multi-Agent Path Finding (abb. MAPF) literature mainly focuses on static environments where agents are the only re-configurable entities [12] [13]. The only exceptions we could find are Bellusci [14] and Vainshtain [15] recent C-MAPF and TF-MAPF problem extensions, where the environment contains movable shelves in warehouses. However, in Bellusci's work, shelves are in fact moved by humans *before* robots only navigate in the environment. On the other hand, in Vainshtain's work, only specific robots can move shelves by passing below them ; also both problems are firmly limited to cell-sized robots and objects in grids.

In this paper we introduce the Multi-Robot NAMO problem and propose reactive strategies to manage conflicts and deadlocks, some of them exploiting the Social-NAMO approach.

III. PROBLEM DEFINITION

Before defining the Multi-Robot NAMO problem, we introduce entities and structures common to NAMO problems considered in this paper :

- W is the physical representation of the world. In this paper, entities $E_i \in W$ are considered undeformable polygons in a 2D plane. Their center and orientation at time t is denoted as their configuration : $q^t_{E_i} = (x^t_{E_i}, y^t_{E_i} \in \mathbb{R}^2, \theta^t_{E_i} \in [0, 2\pi[)$.
- R is the set of mobile robots, considered as identical in this paper. $R =< R_1, ..., R_r > \subset W$, for which we only consider the 3 degrees of freedom of their base.
- S is the set of static obstacles, ie. that cannot move nor be moved. $S =< S_1, ..., S_s > \subset W$, defining the layout of the environment (eg. walls and heavy furniture like desks),
- M is the set of movable obstacles that can be moved by robots. $M =< M_1, ..., M_m > \subset W$, $F \cap M = \emptyset$ (they can have different shapes).

A navigation plan p_{R_i} followed by a robot is made of path components. A path component is a sequence of robot configurations $q^t_{R_i}$ where the robot moves along. A path component is of two types :

- (*transfer path*), where the robot moves with a **single object** (with a fixed transform)
- (*transit path*), where the robot moves alone between two configurations.

Definition 1 (MR-NAMO problem): Given an initial workspace configuration W with a set of robots R and a set of movable obstacles M, the MR-NAMO problem consists in computing, if they exist, for each robot R_i a collision-free plan p_{R_i} from its initial configuration $q^{t_{init}}_{R_i}$ to a goal configuration $q^{t_g}_{R_i}$, where moving obstacles in M is allowed.

An optimal NAMO plan would first minimize the number of transferred obstacles, then the transferred distance by the robot, according to Stilman [1]. In MR-NAMO, minimiz-

ing these costs remains of course of primary interest, but optional.

The inherent problem of multi-robot coordination is of resource conflicts [12]. These conflicts arise when a pair of single-agent plans (p_{R_i}, p_{R_j}) require concurrent access to resources that cannot be shared. In the MR-NAMO problem, two resource types are shared: space and movable obstacles.

Space Conflicts arise when plans (p_{R_i}, p_{R_j}) require the robots or the transferred obstacles to reach intersecting configurations $(q^t_{E_k}, q^t_{E_l})$ where ${E_k}^t \cap {E_l}^t \neq \emptyset$.

Movable Obstacles Conflicts, arise when plans (p_{R_i}, p_{R_j}) require the robots to manipulate a same obstacle M_k.

In the next section we examine how to detect and avoid such conflicts while defining implicit approaches to deal with deadlock situations.

IV. IMPLICIT COORDINATION

A. Overview

As Yan *et al.* explain in [12], implicit coordination only relies on the robot's perceived knowledge about the world, and takes reactive/local action to adapt its plan of execution in response to other robots' actions. It thus avoids the exponential complexity of searching the space of all robot configurations during planning. Moreover, implicit coordination does not require direct communication between robots, it provides solutions in any environment, even with poor communication conditions. These advantages come at the cost of completeness and optimality, but make the problem tractable, even with limited computational resources.

We present an implicit coordination strategy for MR-NAMO which is summarized in pseudo-code in Algorithm 1. The algorithm requires robots to independently plan and locally react to conflicts, ensuring that planning time linearly increases with the number of robots. It can use any of the existing single-robot NAMO planners as a subroutine (in our case, either NAMO or S-NAMO). We now provide a summary of the algorithm before describing two of its key features, *conflict avoidance* and *deadlock resolution*, in greater detail in sections IV-C and IV-D.

B. Algorithm Summary

Algorithm 1 draws inspiration from the Fixed-path Coordination described by S. Lavalle in his reference book on Motion Planning [16]. This involves planning for each robot using a single-robot algorithm, then scheduling the motion of robots as to prevent conflicts, by either tuning motion speed or introducing pauses. In our case, explicit scheduling is replaced with reactive conflict avoidance and deadlock resolution strategies.

Algorithm 1 is executed by each robot at each time step. First, the robot checks if the goal has been reached (L2). Then it checks if the current plan is empty, and if so, it computes a new plan (REPLAN L3). Next, the robot calls **DETECT_CONFLICTS** (L4) which returns a set of conflict objects encapsulating the type of conflict and entities involved. The robot caches all conflicts seen while pursuing

Algorithm 1 Overall Strategy - Determines the next agent action based on the current world state and updates the agent's plan as needed. In blue, deadlock resolution.

Input: The current world state W, robot pose q_r, goal pose q_g, current plan p (if any), and number of look ahead steps h.
Output: The next agent action and the updated plan.

```
1:  procedure OVERALL_STRATEGY(W, q_r, q_g, p, h)
2:      if q_r = q_g then return SUCCESS, p
3:      if p = ∅ then return REPLAN(W, q_r, q_g, h)
4:      C ← DETECT_CONFLICTS(p, W, h)
5:      if C then
6:          D ← DETECT_DEADLOCKS(C)
7:          if D then
8:              p ← EVADE_OR_WAIT(D, p, W, q_g)
9:          else if REQUIRES_REPLAN(C) then
10:             return REPLAN(W, q_r, q_g, h)
11:         else
12:             p ← POSTPONE(p)
13:     return NEXT_ACTION(p), p
14: end procedure
15:
16: procedure REPLAN(W, q_r, q_g, h)
17:     W' ← copy of W with all agents removed.
18:     p ← BASE_NAMO_PLANNER(W', q_r, q_g)
19:     if p = ∅ then return FAIL, p
20:     C ← DETECT_CONFLICTS(p, W, h)
21:     if C then
22:         W'' ← copy of W' with conflicting agents
                added as static obstacles.
23:         p' ← BASE_NAMO_PLANNER(W'', q_r, q_g)
24:         C ← DETECT_CONFLICTS(p', W, h)
25:         if p' = ∅ OR C then p ← POSTPONE(p)
26:         else p ← p'
27:     return NEXT_ACTION(p), p
28: end procedure
```

the same goal for later deadlock detection. If conflicts are detected, the robot calls **DETECT_DEADLOCKS** to examine them for deadlocks, and if any are found, it either evades or postpones (L8). If conflicts are detected without any deadlocks, the robot either postpones or replans immediately (L9-12). Finally, the robot executes the next action in its plan (L13).

In the **REPLAN** subroutine (L16-28), the single-robot NAMO planner is first called to find a plan ignoring all other robots (L17-18). If no plan is found, the goal must fail because the problem is unsolvable for the NAMO planner. If the plan has conflicts (L20), we try to recompute a plan by considering the conflicting entities as static obstacles (L22). If *that* plan is empty or has conflicts, then the plan that ignored other robots is chosen and postponed for a random duration (POSTPONE L25). Finally, to avoid infinite replanning, a BASE_NAMO_PLANNER call counter is kept for each goal and causes goal failure at a threshold.

C. Conflict Avoidance (CA)

As stated in Section III, conflicts represent incompatibilities between robots' plans. In implicit coordination, since plans are not shared, detecting actual conflicts is not possible; but they can be predicted with more-or-less accuracy. This is why we first define *Potential Conflicts*, or observable situations that could become actual Space or Movable Obstacle

conflicts, if no action is taken to avoid them. But for the sake of readability, we will henceforth refer to potential conflicts as conflicts.

This results in a total of six types of conflicts, as shown in Fig. 2. **Robot-Robot** conflicts occur when the robot's plan intersects with another robot (or the obstacle it is transferring), within a horizon of a fixed number of steps h. **Object in Path** conflicts occur when a movable obstacle is left intersecting with the robot's plan. **Simultaneous Space Access** conflicts are a particular case of Robot-Robot conflict where two or more robots are close enough they could collide within the next time step. **Stealing Object** occurs when the robot planned to move an obstacle, but another is currently transferring it. **Stolen Object** occurs when the robot planned to move an obstacle, but it has already been moved and released by another robot. **Simultaneous Grab** occurs when the robot is within one step from manipulating an obstacle as planned, but another robot is close enough it could also grab within the same time step.

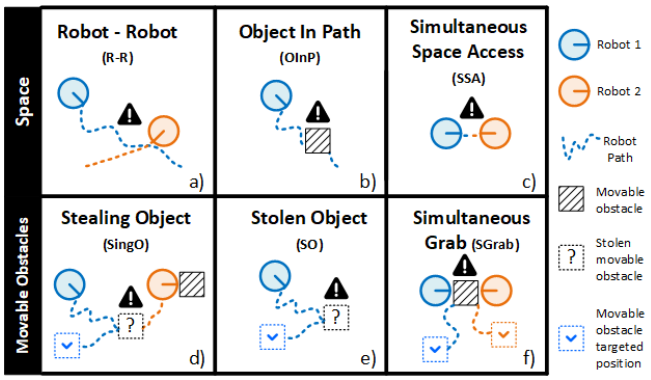


Fig. 2: Potential conflicts classification in MR-NAMO

Upon detecting potential conflicts, Algorithm 1 proceeds with *conflict avoidance* which amounts to either replanning or postponing the robot's plan for a random number of steps. The randomization breaks symmetrical conflicts like Simultaneous Space Access and Simultaneous Grab. For Robot-Robot conflicts, postponement allows the crossing robot to clear the path, whereas for Stealing Obstacle, it provides time for the thief to move the obstacle. Immediate replanning is necessary for conflicts where no other robot is directly involved, such as Object in Path and Stolen Object which are unlikely to be resolved after waiting (Alg. 1 REQUIRES_REPLAN L9).

D. Deadlock Resolution (DR)

Implicit coordination can not guarantee the resolution of all deadlock situations, defined as dependency cycles between robots' actions [17]. These typically result in robots freezing or oscillating between two configurations indefinitely. Similarly to conflicts, detecting actual deadlocks requires the sharing of robot plans. Thus, without explicit communication, robots can only detect *potential* deadlocks.

We define a potential deadlock as the repetition of a robot-robot potential conflict over time: more precisely, if a robot detects a conflict with all involved robots in the exact same configurations, within a single goal navigation's time span, a potential deadlock is detected (Alg. 1 L6). To break the cycle, the robots must decide which of them should move to give way to the others, without explicitly communicating (Alg. 1 EVADE_OR_WAIT L8).

Our first method of deadlock resolution, which we call **Repulsive Deadlock Resolution** (Repulsive DR), seeks to maximize distance between robots in an attempt to break the cycle. In Repulsive DR, we operate an arbitrary selection : the robot with the smallest position vector (i.e. $x^2 + y^2$) chooses to postpone (*postponer*) while the others evade (*evaders*). The evaders perform an A* grid search for an evasion cell that is maximally distant from the other robots. The search ends after a fixed number of cells have been visited. This value is a hyper-parameter which we set to 1000 in experiments. The postponer robot waits for a random number of steps within a predefined interval.

As we will show in section V, Repulsive DR, while simple and effective, greatly increases the average distance traveled and fails to take advantage of map topology. Fig. 3a shows an example where it fails to resolve a deadlock which could be solved by the lower-cost (closer) evasion position shown in Fig. 3b. Moreover, the choice of evader is crucial when robots are unequally constrained in space. The wrong evader selection will fail to end deadlocks that require a certain robot to move first. For example, in Fig. 3c, Robot 2 will fail to reach its goal after evading.

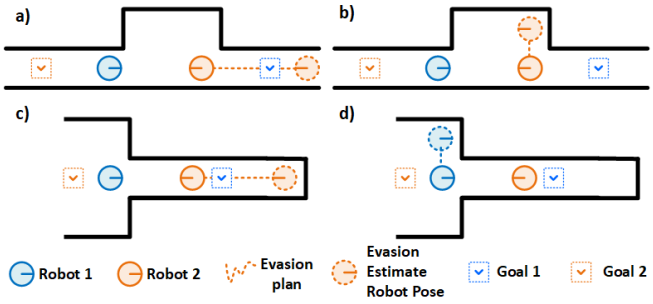


Fig. 3: Repulsive DR failure cases[1]: a) highlights a robot's repulsive evasion away from the other robot, b) illustrates an ideal evasion behavior on this configuration. c) illustrates a bad evader selection resulting in an evasion failure, where d) shows an ideal behavior with a correct evader choice.

To improve the selection of evaders and evasion configurations, we propose a second evasion strategy called **Social Deadlock Resolution** (Social DR). This is used exclusively with the S-NAMO base planner because it takes advantage of the social costmap [2] (illus. Fig. 4.a). In Social DR, an A* grid search is performed to find an evasion cell with minimum *compromise-cost*, which is a weighted average of the normalized distance and social cost. More precisely, the distances and social costs of all explored evasion cells are normalized to range between 0 and 1 and the compromise-

[1]Video of our work : https://youtu.be/oi3iM8AmpuY

cost $g(x, y)$ is given by:

$$g(x, y) = \frac{w_d \cdot d(x, y) + w_{sc} \cdot sc(x, y)}{w_d + w_{sc}}$$

Where $d(x, y)$ is the normalized distance, $sc(x, y)$ is the normalized social-cost, and w_d and w_{sc} are hyper-parameters respectively weighing the travel distance against the social cost at the evasion configuration. When $w_d > w_{sc}$, the algorithm will favor nearby evasion configurations, and conversely, when $w_d < w_{sc}$, it will favor better evasion configurations with regards to social cost.

The robot with the highest social cost evasion cell becomes the postponer, while the others become evaders. The search limit hyper-parameter remains the same as in Repulsive DR. Fig. 4 illustrates this process.

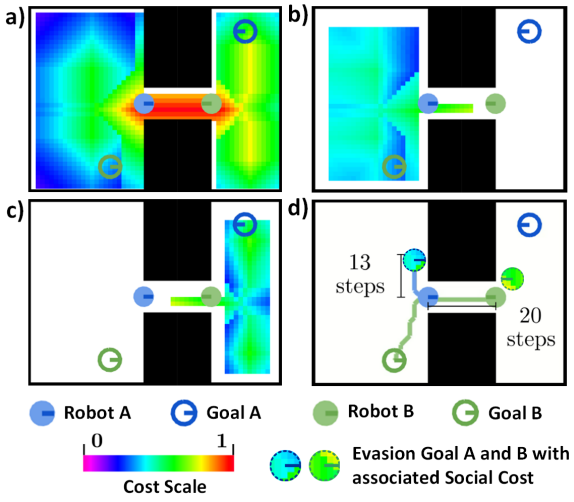


Fig. 4: Social DR: The social costmap (a) is pre-computed and cached by each robot at the start of the simulation. During deadlock resolution, each involved robot independently computes A* grid searches to find an evasion cell with the lowest *compromise-cost* for themselves and for the others. The robot with the highest social-cost evasion cell postpones while the others evade. (b) Compromise cost for robot A. (c) Compromise cost for robot B. In (d), each robot's evasion goal is evaluated, robot A chooses the evasion cell of lower social-cost and executes its evasion plan while robot B postpones.

V. EXPERIMENTAL EVALUATION

A. Experimental Context

We implement the implicit coordination strategy by extending our open-source simulator[2], which considers only 2D-polygonal geometry for interaction (no kinematics nor dynamics) akin to the one used by Stilman *et al.* in [1]. The simulator provides an integration layer with ROS for visualization in Rviz and synchronizes timesteps across robots such that varying planning times and randomized postponement durations do not affect the determinism of planning. This makes experiments reproducible, given that the random seed is saved as a scenario parameter.

[2]All code and data is available at:
https://gitlab.inria.fr/chroma/namo/namosim

We perform a set of experiments on the implicit coordination strategy with and without Conflict Avoidance (CA) and Deadlock Resolution (DR) enabled in order to analyze their impact. We also evaluate both the NAMO (Stilman baseline) and S-NAMO base algorithms to highlight the advantages and disadvantages of the social cost model and social evasion method of deadlock resolution. When DR is enabled, the NAMO algorithm uses repulsive evasion (Repulsive DR) and S-NAMO uses social evasion (Social DR). Note that if DR is activated then so is CA, but not the inverse. Thus, there are three variants each for NAMO and S-NAMO, for a total of six algorithms, as shown in Fig. 5f.

In total, we evaluate all six algorithm variants across twenty randomized simulations on the two environments depicted in Fig. 6. Experiments are repeated with a static number of robots ranging from 1 to 10, where each robot has a sequence of 50 navigation goals. The total number of simulations per environment is thus $6 \cdot 20 \cdot 10 = 1200$. The results of these experiments are visualized in Fig. 5 and summarized numerically in Table I. All simulations were executed on a single machine with an AMD Ryzen 9 7950X 16-Core Processor (4.5GHz) and 64GB of RAM, and distributed evenly across CPU cores for fair comparison of planning times.

B. Scenarios

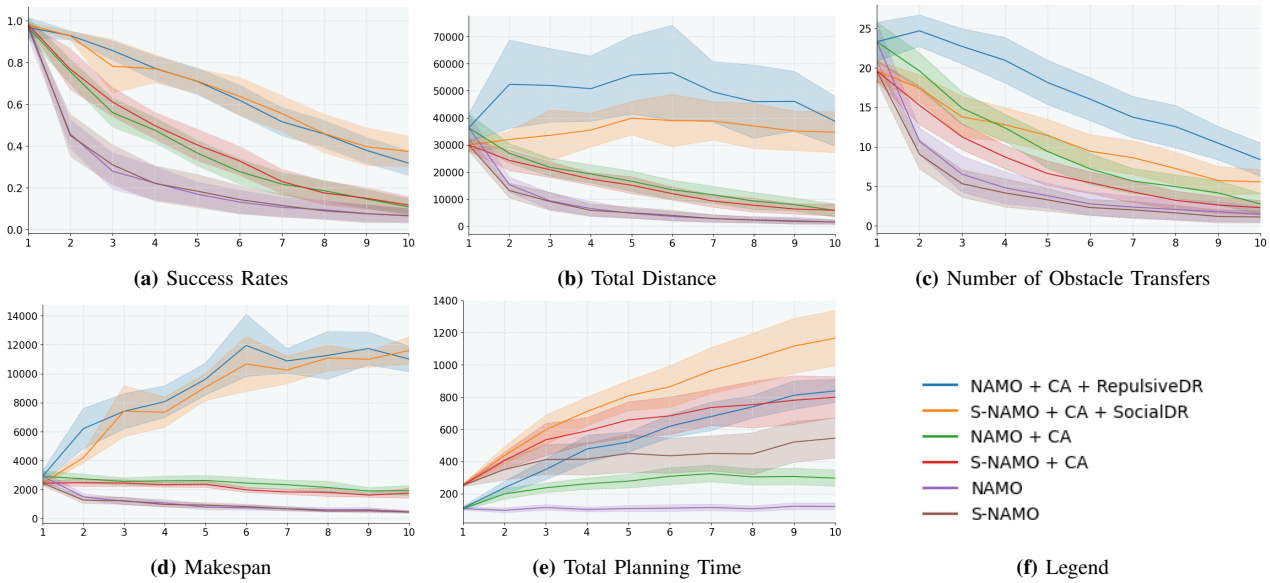
We compare our approaches on the two scenarios illustrated in Fig. 6. The "Intersections" scenario (abb. Int) is designed specifically with multiple intersections next to each other, since we have identified in [2] that they are particularly challenging for NAMO algorithms. The "Willow Garage" scenario (abb. WG) is based on a subsection of the well-known Willow Garage map. It demonstrates the potential use of our NAMO algorithms in a real office environment with a combination of small rooms. In this scenario, we also highlight the ability of the simulator to deal with heterogeneous movable obstacles of varying shape and orientation.

Variations of these two scenarios are generated by randomizing the initial robot poses and goal poses. Because objects must be placed in relevant locations to create actual NAMO problems, their positions are not randomized but set manually and irregularly. The maximum number of BASE_NAMO_PLANNER calls is set to 20 and a random timer duration for postponements is picked within an interval of 5 to 20 steps.

For social DR, we set $w_{sc} = 3$ and $w_{sc} = 2$, as to relevantly favor social evasion configurations. h is set to 10 (simulation steps), as to allow sufficient spatial clearance around robots to efficiently solve conflicts. Freezing these parameters allows us to focus our experiments on the number of robots.

C. Evaluation

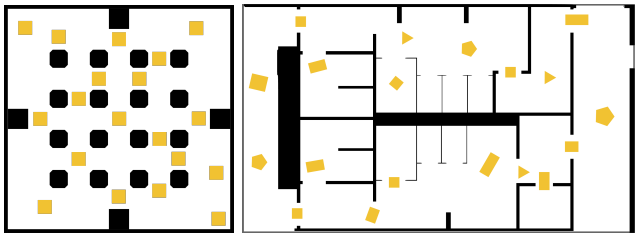
Statistics are accumulated for each robot at each simulation step. Among them, we count the number of obstacle transfers (each time an obstacle is released), the goal success



(a) Success Rates **(b)** Total Distance **(c)** Number of Obstacle Transfers

(d) Makespan **(e)** Total Planning Time **(f)** Legend

Fig. 5: Simulation Results on the "Intersections" Scenario - Each point represents an average per agent per simulation over 20 randomized simulations. An agent either succeeds or fails its current goal before moving on to the next and repeats until all are 50 goals finished. The x-axis represents the number of agents in the simulation, all of which are identical.



(a) Intersections (abb. Int, ∼9.5x9.5m) **(b)** Willow Garage center (abb. WG, ∼9x15m)

Fig. 6: Base environments for scenarios. Static obstacles in black, movable in yellow.

rate (succeeded / total), total planning time (in seconds), and the total distance traveled throughout the simulation (in meters). We also calculate the *makespan* metric which is the number of simulation steps before all robots complete (or fail) all of their objectives. These results are plotted for the Intersections scenario in Fig. 5 and summarized for both environments in Table I. All values provided in the graphs and table represent a per-robot average over 20 randomized simulations.

D. Results and Analysis

Results from the Intersections scenario provide the most relevant information on "worst case" behavior in MR-NAMO, because the presence of many intersecting passages and movable obstacles increases the probability of conflicts. These results are plotted in Fig. 5. We find similar trends on all other scenarios tested, suggesting that data from the Intersections scenario is representative of general performance. The results on the Willow Garage scenario, which is more realistic, are available in Table I.

a) Algorithm performance in MR-NAMO: Fig. 5a shows the average goal success rate per agent in the Intersections environment for each algorithm variant. Here,

Scen.	nb. Rob.	Method	Succ. Rate	Dist.	nb. Transf.	makesp.	Plan. time
Int.	1	NAMO	0.97 ±0.10	36292 ±10248	23.3 ±4.9	2908 ±744	111 ±18
		S-NAMO	0.98 ±0.02	29988 ±4352	19.6 ±2.6	2429 ±316	252 ±16
	5	NAMO Repulsive DR	0.71 ±0.13	55797 ±28791	18.2 ±5.6	9620 ±2202	520 ±124
		S-NAMO Social DR	0.71 ±0.12	39871 ±12394	11.4 ±4.1	9066 ±1843	807 ±186
	10	NAMO Repulsive DR	0.32 ±0.12	38705 ±18316	8.4 ±4.2	11009 ±1746	837 ±144
		S-NAMO Social DR	0.37 ±0.15	34685 ±15010	5.6 ±3.1	11596 ±1868	1166 ±344
WG	1	NAMO	0.97 ±0.03	44013 ±8393	29.6 ±5.75	3645 ±626	201.69 ±52.16
		S-NAMO	0.97 ±0.03	49611 ±10951	18.8 ±2.00	3887 ±783	348.27 ±48.13
	5	NAMO Repulsive DR	0.81 ±0.11	100770 ±35176	15.43 ±4.88	13574 ±3187	625.46 ±144.16
		S-NAMO Social DR	0.73 ±0.15	61100 ±19603	8.31 ±3.21	9159 ±1524	784.83 ±417.34
	10	NAMO Repulsive DR	0.42 ±0.14	74430 ±29139	8.19 ±3.65	14746 ±2615	988.50 ±176.96
		S-NAMO Social DR	0.42 ±0.17	53215 ±26582	4.67 ±2.52	13595 ±3452	1145.21 ±544.19

TABLE I: Average robot performance metrics over 20 randomized simulations on two environments. Results are shown for NAMO and S-NAMO, each with CA and DR activated, on Intersections (Int.) and Willow Garage (WG).

we can clearly see the benefit of using CA and DR. With a simple CA, algorithms managed to improve the success rate by 44%, on average. Moreover adding DR to CA algorithms drastically increases the success rate by 43% on average and up to 67% for 10 robots. Indeed, as the number of robots increases, dealing with complex cyclic robot behaviour becomes mandatory to reach robot goals.

As expected, since only implicit coordination is used, the success rate decreases as the number of robots increases, for all variants. Furthermore, as the number of conflicts and their resolution increases, the robots need more steps to complete their task (makespan, Fig. 5d). The makespan values for

algorithms with CA and DR remain close regardless of the scenario type (Table I). On the other hand, for algorithms without CA, the makespan remains stable with the number of robots. This is mainly due to the low success rate of such algorithms. These trends hold in both environments.

b) Total Distance: Minimizing energy expenditure remains a key objective in most robot navigation problems. Resolving conflicts and deadlocks requires additional displacement cost, which explains why the blue and yellow curves of Fig. 5b with CA+DR show the highest average total distance traveled per agent. Note that as the success rate decreases so do the number of goals reached and thus also the total distance covered. We also observe that using Social DR yields on average a 25% (29% in WG, see Table I) reduction in distance traveled relative to Repulsive DR. This is mainly due to Social DR relying on the compromise-cost to select an evasion position, whereas Repulsive DR seeks to evade as far as possible from the other robot.

c) Obstacle Transfers: A robot's ability to reliably move obstacles is assumed in our simulations. However, obstacle manipulation remains a challenging and risky task for real robots, as it can lead to unrecoverable failures and damage to the robot, obstacle, and environment. Consequently, for the task of navigation, obstacle transfers should generally be avoided, depending on the cost of alternative paths. We find that S-NAMO significantly reduces the average number of obstacle transfers per agent and that this benefit holds as the number of robots increases. This is true with all variants, and shows up to a 40% reduction from Repulsive DR to Social DR. This important advantage is a direct result of S-NAMO seeking to place obstacles in positions of low social-cost where they are less likely to degrade space connectivity, thereby keeping passageways clear.

d) Planning Time: CA and DR incur additional computation and hence planning time, as is evident in Fig. 5e where the two curves with CA+DR (blue and yellow) are higher than the others. They also have the steepest slopes, as more robots leads to more conflicts and deadlocks, which cause more frequent calls of CA and DR subroutines. On the other hand, the two curves with neither CA nor DR (purple and brown) remain nearly constant with increasing numbers of robots. We further observe that S-NAMO incurs a constant increase in planning time relative to NAMO, as is evident from the fact that the group of S-NAMO curves appears offset by a constant amount from the NAMO curves.

VI. CONCLUSION

We have introduced the novel problem of MR-NAMO which merges NAMO and Multi-Robot Coordination. We have also presented a generic implicit coordination algorithm which can be used to extend any single-robot NAMO planner with minimal requirements. To break deadlocks, we introduced a simple repulsive strategy and a more elaborate social-cost compromise strategy (based on S-NAMO [2]). We have evaluated the coordination strategy using both a Stilman baseline NAMO algorithm and the S-NAMO variant. The simulated experiments indicate that 1) our coordination strategy effectively resolves conflicts and deadlocks, and 2) the S-NAMO approach maintains a comparable goal success rate as plain NAMO while exhibiting notable advantages, including fewer obstacle manipulations, reduced total distance, and decreased makespan, albeit with a constant increase in computation time. Furthermore, our open-source simulator, scenarios, and data constitute a first benchmark for the MR-NAMO problem.

For further work, our research will concentrate on integrating or developing additional NAMO algorithms, such as Wu&Levihn's [18], to handle partial environmental knowledge, thereby enhancing real-world applicability. Additionally, we aim to establish local optimality in simple NAMO cases and explore machine learning methodologies.

REFERENCES

[1] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 02, no. 04, pp. 479–503, 2005.

[2] B. Renault, J. Saraydaryan, and O. Simonin, "Modeling a Social Placement Cost to Extend Navigation Among Movable Obstacles (NAMO) Algorithms," in *IROS 2020 - IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Las Vegas, US, 2020, pp. 11 345–11 351.

[3] Renault, Benoit, Saraydaryan, Jacques, and Simonin, Olivier, "Towards S-NAMO: Socially-aware Navigation Among Movable Obstacles," in *Robot World Cup XXIII*, ser. LNCS. Sydney: Springer, 2019.

[4] K. Ellis, H. Zhang, *et al.*, "Navigation among movable obstacles with object localization using photorealistic simulation," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2022, pp. 1711–1716.

[5] J. Muguira-Iturralde, A. Curtis, *et al.*, "Visibility-aware navigation among movable obstacles," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 10 083–10 089.

[6] E. Mueggler, M. Faessler, *et al.*, "Aerial-guided navigation of a ground robot among movable obstacles," in *IEEE Int. Symp. on Safety, Security, and Rescue Robotics (2014)*, Oct. 2014, pp. 1–8.

[7] L. Lindzey, R. A. Knepper, *et al.*, "The Feasible Transition Graph: Encoding Topology and Manipulation Constraints for Multirobot Push-Planning," in *Algorithmic Foundations of Robotics XI: Sel. Contrib. of the 11th Int. Workshop on the Algo. Found. of Robotics*, ser. Springer Tracts in Advanced Robotics. Springer Int. Pub., 2015, pp. 301–318.

[8] M. Levihn, T. Igarashi, and M. Stilman, "Multi-robot multi-object rearrangement in assignment space," in *2012 IEEE/RSJ International Conf. on Intel. Robots and Systems*, Oct. 2012, pp. 5255–5261.

[9] K. Brown, O. Peltzer, *et al.*, "Optimal Sequential Task Assignment and Path Finding for Multi-Agent Robotic Assembly Planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 441–447.

[10] C. R. Garrett, R. Chitnis, *et al.*, "Integrated Task and Motion Planning," *Annual Review of Control, Robotics, and Auto. Systems*, vol. 4, 2021.

[11] J. Motes, R. Sandström, *et al.*, "Multi-Robot Task and Motion Planning With Subtask Dependencies," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3338–3345, Apr. 2020.

[12] Z. Yan, N. Jouandeau, and A. A. Cherif, "A Survey and Analysis of Multi-Robot Coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[13] R. Stern, N. R. Sturtevant, *et al.*, "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," *SOCS*, 2019.

[14] M. Bellusci, N. Basilico, and F. Amigoni, "Multi-Agent Path Finding in Configurable Environments," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2020, pp. 159–167.

[15] D. Vainshtain and O. Salzman, "Multi-agent terraforming: Efficient multi-agent path finding via environment manipulation," *Proc. of the Int. Symp. on Combinatorial Search*, vol. 12, no. 1, pp. 239–241, 2021.

[16] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

[17] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Pub. Co., Inc., 1999.

[18] M. Levihn, M. Stilman, and H. Christensen, "Locally optimal navigation among movable obstacles in unknown environments," in *IEEE-RAS*, 2014, pp. 86–91.