Neuro-Symbolic Deadlock Resolution in Multi-Robot Systems

Ruiyang Wang RUIYANG.WANG@DUKE.EDU
Bowen He BOWEN.HE@DUKE.EDU
Miroslav Pajic MIROSLAV.PAJIC@DUKE.EDU
Duke University, NC, Durham, 27708

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

This work addresses the problem of deadlock situations that are common in decentralized multi-robot missions. Existing approaches focus on predicting potential deadlocks and intervening before they actually occur. However, these methods often struggle to detect all possible deadlocks, especially in environments with uncontrollable obstacles, and may inevitably introduce new deadlocks after interventions. Consequently, we propose a neuro-symbolic deadlock resolution (NSDR) method based on Neural Logic Machines (NLMs). NSDR is designed specifically to resolve deadlocks after their occurrence, with the guarantee that no further persistent deadlocks will emerge after the initial resolution in environments with or without obstacles. Our approach leverages the similarity in logic rules when resolving simple deadlocks involving a small number of robots; this facilitates their use when resolving more complex scenarios with larger robot groups. Training NSDR on simpler deadlock cases allows it to generalize and effectively resolve more complex situations by utilizing the logic rules it has learned from simple deadlocks. We thoroughly evaluate the method in case studies with varying numbers of robots involved in deadlocks and show that NSDR outperforms the state of the art methods, which are based on the use of the adaptive repulsive force.

Keywords: Multi-robot Planning, Neural-Symbolic Artificial Intelligence, Deadlock Resolution

1. Introduction

Deadlock situations in multi-robot systems have attracted significant attention in the field of multi-robot motion planning as discussed in Yin et al. (2024); Luis et al. (2020); Park et al. (2022). These situations arise when a group of robots try to navigate through an obstacle-dense environment, such as passing through a narrow corridor, while avoiding each other, as illustrated in Fig. 1. When a deadlock occurs, the control policy of each robot, e.g., Model Predictive Control (MPC) as in Tallamraju et al. (2018); Arul et al. (2023) or Reinforcement Learning (RL)-based controllers as in Chen et al. (2017); Fan et al. (2020), will force each robot to stop moving in order to remain safe. Meanwhile, the multi-robot system under deadlock cannot make any progress to its assigned targets without outside interventions. Deadlock occurrences can be predicted and thus prevented from a *centralized* infinite-horizon controller as in Tang et al. (2018), but such a controller may not be available or may be computationally infeasible for missions with a large number of robots.

Recent works aim to predict and prevent deadlocks in multi-robot systems with decentralized control. Some methods analyze the force equilibrium of robots during deadlocks; e.g., Grover et al. (2021) derive a deterministic control policy that can drive the multi-robot systems out of the deadlock situation without incurring future deadlocks. Yet, this method has only been proven to work for scenarios with two and three agents in *obstacle-free* environments due to the complexity of geometric properties that the method uses in more complicated systems. Chen et al. (2024)

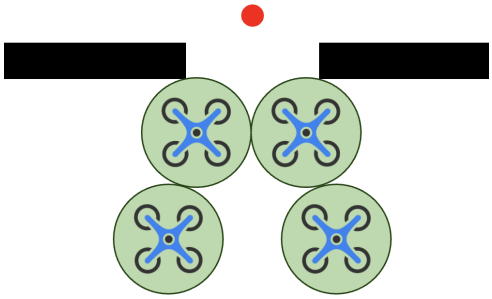


Figure 1: Deadlock occurs as robots navigate through a narrow corridor to the red target.

extend this approach, showing that deadlocks caused solely by controllable robots are inherently unstable and can be resolved by increasing repulsive forces among agents before an equilibrium is reached. However, when obstacles are introduced Chen et al. (2023), this guarantee is no longer valid, as static obstacles cannot be controlled. In response, an importance-based greedy method assigns the highest importance to one robot to repel others by amplifying its repulsive force in Chen et al. (2023). Nonetheless, this method lacks guarantees for deadlock resolution in the presence of obstacles, and the system may still experience deadlocks, as we show in this work.

To the best of our knowledge, no method exists that can resolve deadlocks after their occurrence with strong formal and performance guarantees. Wang et al. (2017) and Toumieh and Lambert (2022) utilize pre-defined perturbations to the robot controllers according to the right-hand rule when deadlocks occur. Zhou et al. (2017), Pierson et al. (2020), and Abdullhak and Vardy (2021) try to find a temporary target point on the right-hand side of robots empirically when deadlocks occur. But the pre-defined nature of these types of methods cannot guarantee the resolution of the current deadlocks nor the prevention of entering future deadlocks. Other potential methods may involve relaxing or removing constraints from the decentralized controllers of each robot in order to recover from a deadlock, as in Lee et al. (2023); Wang et al. (2022); Parwana et al. (2024); however, these methods cannot guarantee the system exit from deadlocks.

Consequently, we first introduce the *Active-Passive* paradigm for deadlock resolution and prove that combining it with a greedy method, as has been done in Chen et al. (2023), ensures feasibility and prevents future deadlocks after resolving the initial one. We then improve the efficiency of deadlock resolution with a neuro-symbolic deadlock resolution (NSDR) method that is based on Neural Logic Machines (NLMs) Dong et al. (2019). NSDR utilizes NLM training on simple deadlocks in scenarios with few robots, and is capable of generalizing logic rules for resolving complex deadlocks using the Active-Passive paradigm; specifically, NSDR exploits the similarity in NLM-obtained logic rules for simple deadlock resolutions with a small number of robots, and more complex scenarios with larger robot groups. Finally, we perform an empirical comparison involving different numbers of robots between the NSDR method and the Greedy method with the Active-Passive paradigm, as well as compare both methods with the adaptive repulsive force method from Chen et al. (2023).

The paper is organized as follows. Sections 2 and 3 present the problem formulation and proposed methods, respectively, and the NSDR's performance is thoroughly evaluated in Section 4. Finally, Section 5 provides concluding remarks.

2. Problem Formulation

2.1. Robot Dynamics and Decentralized MPC Controller

Consider a set of M homogeneous robots, $\mathbb{M} = \{m^i | i = 1, ..., M\}$, where each robot's safety regions is represented as a ball in $\mathbb{R}^2$ with radius r centered at position p^i. The discrete dynamics of a robot may be approximated by a double integrator:

$$x^i_{t+1} = \mathbf{A}x^i_t + \mathbf{B}u^i_t, \tag{1}$$

where $x^i_t = \begin{bmatrix} p^i_t \\ v^i_t \end{bmatrix}$ is the state (position p^i_t and velocity v^i_t) and u^i_t is the control input (acceleration). Here, $\mathbf{A} = \begin{bmatrix} \mathbf{I}_2 & \delta_t \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \mathbf{0}_2 \\ \delta_t \mathbf{I}_2 \end{bmatrix}$, with $\mathbf{I}_2$ as the 2x2 identity matrix, and δ_t as time difference between consecutive timesteps. Physical constraints on maximum velocity and acceleration are modeled as $\mathbf{I}_2|v^i_t| \leq v_{max}$ and $\mathbf{I}_2|u^i_t| \leq a_{max}$, where $|\cdot|$ is the element-wise absolute value.

Given a list of N static obstacles $\mathbb{O} \triangleq \{o^j | \forall j \in \{1, ..., N\}\}$, where each static obstacle is a polytope that can be represented as a set of points $\mathcal{P}^j_{obs} \triangleq \{p \in \mathbb{R}^2 | \mathbf{A}^j_{obs} p \leq b^j_{obs}\}$. For a circular robot i with radius r, we can define a set of points $\mathcal{P}^i_{rob,t} \triangleq \{p \in \mathbb{R}^2 | \ ||p - p^i_t||^2_2 \leq r\}$ to enclose all interior points of the robot at timestep t and define a polytope outer-approximation of the $\mathcal{P}^i_{rob,t}$ using $[\mathbf{A}^i_{rob,t}, b^i_{rob,t}] \triangleq \{\mathbf{A}^i_{rob,t}, b^i_{rob,t} | \forall p \in \mathcal{P}^j_{rob,t}, \mathbf{A}^i_{rob,t} p \leq b^i_{rob,t}\}$. Thus, the collision avoidance between robot i at position p^i_t and obstacle j at timestep t can be enforced with constraint $\max(\mathbf{A}^j_{obs} p^i_t - b^j_{obs}) \geq r$ as proved in Paparusso et al. (2024) Proposition 2. Similarly, we can define the constraint for collision avoidance between two robots so that the robot i at position p^i_t will not collide with robot k at timestep t if $\max(\mathbf{A}^k_{rob,t} p^i_t - b^k_{rob,t}) \geq r$, where $\mathbf{A}^k_{rob,t}$ and $b^k_{rob,t}$ is the outer-approximation of robot k at timestep t.

The optimal control input $u^i_t, u^i_{t+1}, \ldots, u^i_{t+T}$ of each robot i starting at timestep t for a time horizon T is then calculated distributively from a constrained optimization problem:

$$u^i_t, u^i_{t+1}, \ldots, u^i_{t+T} = \underset{u^i_{t+\tau} \in U}{\arg\min} \sum\nolimits_{\tau=0}^{T} ||x^i_{\tau+1+t} - x^i_{target}||^2_2 + \lambda ||u_{t+\tau}||^2_2 \tag{2a}$$

$$x^i_{t+\tau+1} = \mathbf{A}x^i_{t+\tau} + \mathbf{B}u^i_{t+\tau} \quad \mathbf{I}_2|v^i_{\tau+1+t}| \leq v_{max} \quad \mathbf{I}_2|u^i_{\tau+1+t}| \leq a_{max} \tag{2b}$$

$$\max(\mathbf{A}^j_{obs} x^i_{t+\tau+1} - b^j_{obs}) \geq r \quad \forall j \in \{1, .., N\} \tag{2c}$$

$$\max(\mathbf{A}^k_{rob,t+\tau+1} x^i_{t+\tau+1} - b^k_{rob,t+\tau+1}) \geq r \quad \forall k \in \{1, .., M\} \setminus i; \tag{2d}$$

here, the set U contains all possible control inputs that a robot can take. As in the literature of receding horizon MPC, robots communicate between each other to exchange their planned trajectories for collision avoidance between robots, and we output u^i_t to the robot, update its state, and repeat the MPC controller at timestep $t + 1$.

2.2. Deadlocks

A deadlock happens at some timestep t_{dead} when

$$||p^i_{t_{dead}+1} - p^i_{t_{dead}}||_2 \leq \epsilon, \quad ||p^i_{t_{dead}+1} - p^i_{target}||_2 \geq \epsilon, \quad \forall i \in \{1, .., M\} \tag{3a}$$

where $\epsilon \geq 0$ is a small constant to verify whether a robot is moving towards its assigned target. This makes the entire multi-robot system in a 'stalemate' in which all robots stay in its current position and thus cannot make any further progress towards their assigned targets. Thus, the objective of this paper is to break the deadlock so that robots to reach their assigned targets as quickly as possible.

Objective *Consider a multi-robot system where each robot employs decentralized controllers using (2), and if a deadlock has occurred as defined in (3), our objective is to:*

1. *Make sure that all robots arrive at their assigned targets at some timestep $t^i_{arrival}$, s.t. $||p^i_{t^i} - p^i_{target}||_2 \leq \epsilon \quad \forall i \in \{1, ..., M\}$ when $t^i = t^i_{arrival}$.*
2. *Minimize the total sum of arrival time for the entire multi-robot system: $J = \sum_{i=1}^{M} t^i_{arrival}$.*
3. *If there are priorities given, we want to minimize the weighted sum of arrival time for the multi-robot system: $J = \sum_{i=1}^{M} \rho^i t^i_{arrival}$, where ρ^i is the priority assigned to the ith robot.*

3. Proposed Methods

3.1. Active-Passive Paradigm

In order to resolve the occurred deadlock, we propose a paradigm of selecting robots to switch from active status to passive status with an initialization that all robots are in active status. Then, a robot i with passive status has the controller as in (4).

$$\textbf{Passive}: u^i_t, u^i_{t+1}, \ldots, u^i_{t+T} = \underset{u^i_{t+\tau} \in U}{\arg\min} \sum_{\tau=0}^{T} ||u^i_{t+\tau}||_2^2 \tag{4a}$$

$$x^i_{t+\tau+1} = \mathbf{A}x^i_{t+\tau} + \mathbf{B}u^i_{t+\tau} \quad \mathbf{I}_2|v^i_{\tau+1+t}| \leq v_{max} \quad \mathbf{I}_2|u^i_{\tau+1+t}| \leq a_{max} \tag{4b}$$

$$\max(\mathbf{A}^j_{obs} x^i_{t+\tau+1} - b^j_{obs}) \geq r \quad \forall j \in \{1, .., N\} \tag{4c}$$

$$\max(\mathbf{A}^k_{rob,t+\tau+1} x^i_{t+\tau+1} - b^k_{rob,t+\tau+1}) \geq r \quad \forall k \in \{1, .., M\} \setminus \{i \cup \mathcal{I}^i_{passive}\} \tag{4d}$$

where $\mathcal{I}^i_{passive}$ is the index set of passive robots known to robot i.

Notice that, for robot i with passive status, we have removed the target arrival objective from its objective function. Thus, the robot controller will be optimized only to avoid collision between robots that have not assigned passive status and static obstacles. As a result, we can remove the collision avoidance constraints for robots with passive status in controllers for robot i with active status as follows:

$$\textbf{Active}: u^i_t, u^i_{t+1}, \ldots, u^i_{t+T} = \underset{u^i_{t+\tau} \in U}{\arg\min} \sum_{\tau=0}^{T} ||x^i_{\tau+1+t} - x^i_{target}||_2^2 + \lambda ||u_{t+\tau}||_2^2 \tag{5a}$$

$$x^i_{t+\tau+1} = \mathbf{A}x^i_{t+\tau} + \mathbf{B}u^i_{t+\tau} \quad \mathbf{I}_2|v^i_{\tau+1+t}| \leq v_{max} \quad \mathbf{I}_2|u^i_{\tau+1+t}| \leq a_{max} \tag{5b}$$

$$\max(\mathbf{A}^j_{obs} x^i_{t+\tau+1} - b^j_{obs}) \geq r \quad \forall j \in \{1, .., N\} \tag{5c}$$

$$\max(\mathbf{A}^k_{rob,t+\tau+1} x^i_{t+\tau+1} - b^k_{rob,t+\tau+1}) \geq r \quad \forall k \in \{1, .., M\} \setminus \{i \cup \mathcal{I}^i_{passive}\} \tag{5d}$$

The controller for robots with active status is identical to the original MPC controller defined in (2) when $\mathcal{I}^i_{passive} = \emptyset$. By switching robots from active status to passive status, we are aiming to select robots that temporarily deviate from their assigned targets and move away from the deadlock position to give space for active robots moving to their assigned targets.

There are many ways we can use to select robots switching from active status to passive status, we first propose a greedy method similar to Chen et al. (2023) based on robots' distance to their assigned targets.

3.2. Greedy Method

When a deadlock occurs, as outlined in Alg. 1, the greedy method first find $m^{i_{closest}}$ as the robot that is closest to its assigned target. Then we order all other robots based on their distance to $m^{i_{closest}}$ and assign them to be passive in decreasing order of their distance to $m^{i_{closest}}$ and update the set of indexes of passive robots $\mathcal{I}_{passive}$ iteratively. The iterative update of $\mathcal{I}^i_{passive}$ also prevent pairs of robots to ignore collision constraints between each other, thus ensuring safety between each robot. After the remaining active robot has arrived at its target, we reactivate passive robots to active status based on the order of their distance to assigned targets one at a time.

Algorithm 1 Greedy Algorithm for Deadlock Resolution

T ; `// Full Time horizon considered`
$\mathcal{M} = \{m^1, \ldots, m^M\}$; `// List of robots initialized with` **`active`** `status`
$\mathcal{I}_{passive} \leftarrow \emptyset$; `// Set of indexes of all passive robots`
for $t = 1$ **to** T **do**
  if *deadlock detected among active robots* **then**
    `// Find the index of the robot closest to its target`
    $i_{closest} \leftarrow \arg\min_{i \in \{1,\ldots,M\}} ||p^i_t - p^i_{target}||_2$
    Sort robots $m^i \in \mathcal{M}$ by decreasing distance to $m^{i_{closest}}$
    `// Update each robot's` $\mathcal{I}^i_{passive}$ `iteratively`
    for *each robot* m^i *in the sorted list of active robots other than* $m^{i_{closest}}$ **do**
      $m^i.\mathcal{I}^i_{passive} \leftarrow \mathcal{I}_{passive} \quad \mathcal{I}_{passive} = \mathcal{I}_{passive} \cup i$
    end
    $m^{i_{closest}}.\mathcal{I}^{i_{closest}}_{passive} \leftarrow \mathcal{I}_{passive}$; `/* Update` $\mathcal{I}^{i_{closest}}_{passive}$ `for` $m^{i_{closest}}$ `*/`
  end
  if *all active robots have arrived at target positions* **then**
    `// Re-activate robots based on the distance to targets`
    $i_{closest} \leftarrow \arg\min_{i \in \mathcal{I}_{passive}} ||p^i_t - p^i_{target}||_2 \quad \mathcal{I}_{passive} = \mathcal{I}_{passive} \setminus i_{closest}$
    Set $m^{i_{closest}}$ status to active $\quad m^i.\mathcal{I}^i_{passive} = m^i.\mathcal{I}^i_{passive} \setminus i_{closest} \; \forall i \in \mathcal{I}_{passive}$
  end
end

For a robot k under dynamics defined in (1), let's define a reachable set with robot initial position at $p^k_{t_0}$ at timestep t_0 in T timesteps:

$$\mathcal{R}^k_{t_0:t_0+T} \triangleq \left\{ p \in \mathbb{R}^2 \mid [\begin{smallmatrix} p \\ v \end{smallmatrix}] = \mathbf{A}^T [\begin{smallmatrix} p^k_{t_0} \\ v^k_{t_0} \end{smallmatrix}] + \sum_{\tau=0}^{T-1} \mathbf{A}^\tau \mathbf{B} u^k_{t_0+\tau}, \; \mathbf{I}_2 |v^k_{t_0+\tau}| \leq v_{max}, \; \mathbf{I}_2 |u^k_{t_0+\tau}| \leq a_{max} \right\}$$

Then a set of polytope outer-approximations of robot k for each position p^k_z inside $\mathcal{R}^k_{t_0:t_0+T}$ is:

$$\mathcal{A}^k_{reach,t_0:t_0+T} \triangleq \left\{ [\mathbf{A}^{k,z}_{rob}, b^{k,z}_{rob}] \mid p^k_z \in \mathcal{R}^k_{t_0:t_0+T}, \; \forall ||p - p^k_z||^2_2 \leq r \; \mathbf{A}^{k,z}_{rob} p \leq b^{k,z}_{rob} \right\}$$

Assumption 1 *For robot i being selected passive status at timestep t_{pass}, there is exist a feasible solution for (4) for all possible pair of $[\mathbf{A}^{k,z}_{rob}, b^{k,z}_{rob}]$ in $\mathcal{A}^k_{reach,t_{pass}:t_{pass}+T}$ in constraint (4d).*

This assumption ensures the passive robot always has feasible trajectories to avoid active robots. The greedy algorithm satisfies this by prioritizing robots furthest from $m^{i_{closest}}$ for passive status, as these boundary robots have the most free space. This creates more room for closer robots, maintaining Assumption 1.

Assumption 2 *All robots are able to arrive at their assigned targets when no other robots present, and robots have arrived at their targets will not affect later robots to arrive at their assigned targets.*

This assumption excludes cases where the target of a robot is inaccessible, such as inside an obstacle, or where the arrived robots block the paths of others.

Theorem 1 *With Assumption 1 and Assumption 2, the Greedy algorithm 1 guarantees that all robots can arrive at their assigned targets without incurring new deadlocks nor infeasibility.*

Proof Base Case: The Greedy algorithm keeps only one robot, $m^{i_{closest}}$, active, with all others assigned as passive. By Assumption 2, $m^{i_{closest}}$ can move towards its target without interference, while Assumption 1 ensures that passive robots can avoid $m^{i_{closest}}$. Thus, $m^{i_{closest}}$ can progress without causing new deadlocks, as it is moving to its assigned target, or infeasibility.

Inductive Step: Once the active robot reaches its target, the Greedy method selects another robot as active. By Assumption 2, previously arrived robots do not block the newly active robot. This process repeats, with one active robot at a time and others in passive status, ensuring that all robots eventually reach their targets without further deadlocks or infeasibility. ■

3.3. NSDR

Although using the greedy method to select robots switching from active to passive status has guarantees in feasibility without new deadlocks, it is highly inefficient as there could only be one robot with active status that move towards its assigned target after the first deadlock detected. We propose a new method, NSDR, based on the Nerual Logic Machines (NLM) in Dong et al. (2019).

The NLM is a neural-based model designed to apply first-order logic rules and draw conclusions based on predicates grounded over a set of objects. A predicate P with arity a, denoted as $P(e_1, \ldots, e_a)$, grounded over a set of robots $\mathcal{M} = \{m^1, ..., m^M\}$ as our interested set of objects, resulting a value tensor $P_t^{\mathcal{M}}$ with the shape of $[M^{\bar{a}}] \triangleq [S^1, S^2, ..., S^a]$ and $S^j = M,\ \forall j \in \{1, ..., a\}$. Each entry $P_t^{\mathcal{M}}(e_{i1}, \ldots, e_{ia})$ has the value indicating whether P is true when the predicate arguments $e_{i1}, \ldots, e_{ia}$ are instantiated by the specific robots $m^1, \ldots, m^a$ from $\mathcal{M}$, i.e., $e_{i1} = m^1, \ldots, e_{ia} = m^a$, with their states at timestep t.

For deadlock resolution, we first design 4 unary predicates, and 3 binary predicates, to describe the state of each robot in relation to others in the system. These predicates are defined as follows:

1. MoveFront(m^i) ← Whether m^i can move in its front direction.
2. MoveBack(m^i) ← Whether m^i can move in ... back ...
3. MoveLeft(m^i) ← Whether m^i can move in its left direction.
4. MoveRight(m^i) ← Whether m^i can move in ... right ...
5. Deadlock(m^i, m^j) ← Whether m^j blocks m^i from moving in any of the four directions.
6. TargetCloser(m^i, m^j) ← Whether m^j is closer to its target than m^i to its own target.
7. HigherPriority(m^i, m^j) ← Whether m^j has a higher priority than m^i.

Thus, we can stack the value tensors from 4 unary predicates to form a new value tensor $P_{t,unary}^{\mathcal{M}}$ with the shape $[M^1, 4] \triangleq [M, 4]$ and stack value tensors from 3 binary predicates to form $P_{t,binary}^{\mathcal{M}}$ with the shape $[M^2, 3] \triangleq [M, M, 3]$. Our key insight is that the NLM perform logic reasoning over different predicates, thus the last dimension of the value tensors is independent of the number

Algorithm 2 NSDR Algorithm for Deadlock Resolution

T ; // Full time horizon considered
$\mathcal{M} = \{m^1, \ldots, m^M\}$; // List of robots initialized with **active** status
$\mathcal{I}_{passive} \leftarrow \emptyset$; // Set of indexes of all passive robots
for $t = 1$ **to** T **do**
  if *deadlock detected among active robots* **then**
    // Evaluate predicate values for $\mathcal{M}$ and obtain ordered list based on NLM scores
    $P^{\mathcal{M}}_{t,unary}, P^{\mathcal{M}}_{t,binary} \leftarrow$ predicate_cal($\mathcal{M}$)
    score_list $\leftarrow$ NLM($P^{\mathcal{M}}_{t,unary}, P^{\mathcal{M}}_{t,binary}$)
    // Select highest feasible scored active robot as $i_{s_{max}}$
    for i ***in*** *score_list* **do**
      if *selection of robot* i *is feasible* ***and*** m^i*.status* == ***active*** **then**
        $i_{s_{max}} \leftarrow i$ **break**
      end
    end
    $\mathcal{I}_{passive} \leftarrow \mathcal{I}_{passive} \cup \{i_{s_{max}}\}$; /* Update $\mathcal{I}_{passive}$ */
    for *each robot* $m^i \in \mathcal{M} \setminus m^{i_{s_{max}}}$ ***and*** m^i*.status* == ***active*** **do**
      $m^i.\mathcal{I}^i_{passive} \leftarrow \mathcal{I}_{passive}$; /* Update $\mathcal{I}^i_{passive}$ for all active robots */
    end
  end
  if *all active robots have arrived at target positions* **then**
    // Re-activate robots based on distance to targets
    $i_{closest} \leftarrow \arg\min_{i \in \mathcal{I}_{passive}} ||p^i_t - p^i_{target}||_2$ $\mathcal{I}_{passive} = \mathcal{I}_{passive} \setminus i_{closest}$
    Set $m_{i_{closest}}$ status to active $m^i.\mathcal{I}^i_{passive} = m^i.\mathcal{I}^i_{passive} \setminus i_{closest}\ \forall i \in \mathcal{I}_{passive}$
  end
end

of robots in $\mathcal{M}$. With this advantage, we can train an NLM on simple scenarios (e.g., with few robots) and generalize to complex situations with more robots, such as larger multi-robot systems by exploiting the similarity in logic rules when resolving deadlocks with different complexities.

The full algorithm of NSDR is outlined in Alg. 2. When a deadlock is detected, we evaluate the value tensors of defined predicates grounded over the multi-robot systems $\mathcal{M}$. The NLM with depth D, as outlined in Fig. 2, takes in the value tensors and outputs unary logical conclusions over the same set of robots by reasoning about the relationships between robots within $\mathcal{M}$ to deduce properties for individual robots. Within the NLM, the value tensors are transformed to different arities using the **Expansion** and **Reduction** operations defined in the original paper Dong et al. (2019) and pass through different Multi-Layer Perceptrons (MLPs) for D iterations. The NLM then generates K unary latent conclusions for each robot, which are then processed by a shared MLP to assign a score for each robot. The robot with the highest score is selected to switch to passive status.

However, as the NLM model could theoretically select any robot to have passive status, we must verify that the selected robot satisfies Assumption 1 and does not cause infeasibility in the controllers of the remaining active robots. We confirm feasibility by evolving the system forward

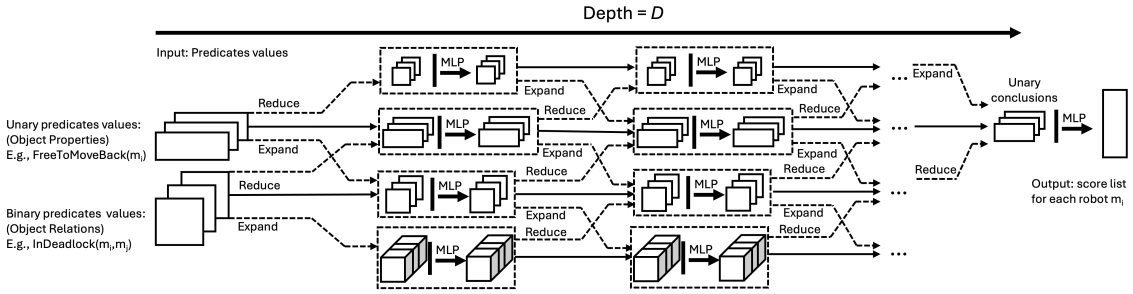


Figure 2: The NLM Pipeline used to select a robot switching to passive status

T_{check} timesteps and guarante feasibility if T_{check} is set to the full planning horizon T. If the highest-scoring robot is infeasible, we move to the next highest-scoring feasible robot. Additionally, because NSDR selects only one robot to switch to passive status when a deadlock is detected, this might not be enough to resolve the initial deadlock, requiring the selection process to repeat until all active robots can advance freely toward their targets. The worst case for NSDR is that it will select $M-1$ robots to be passive and the remaining active robot is able to arrive at its assigned target under Assumption 2. After the initial deadlock resolved with all active robots arrived at their assigned targets, passive robots are reactivated one at a time, prioritized by their distance to their targets, following the same procedure as the greedy algorithm to prevent infeasibility and avoid introducing new deadlocks.

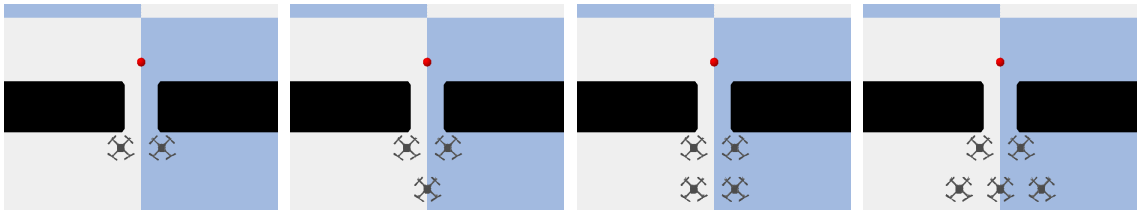
4. Results

We now demonstrate the effectiveness of our algorithms in resolving deadlocks (Fig. 3). A team of Crazyflie robots (modeled with radius $r = 0.06$ m) must navigate a narrow corridor with width of 0.10 m between two rectangular obstacles, where they have entered a deadlock as defined in (3). Each robot's interior is enclosed by a square with side length $d = 2(r + d_{buffer})$, where d_{buffer} is set to 0.10 m to maintain separation between robots. Maximum velocity and acceleration are 0.1 m/s and 0.1 m/s², respectively, with a timestep $\delta t = 0.1$ s. Simulations were conducted in PyBullet for $T = 1000$ timesteps, using CVXPY and the MOSEK solver for the MPC controllers defined in (5) and (4). Upon reaching the target, each robot lands to avoid obstructing others, ensuring Assumption 2 holds. We set $T_{check} = 5$ timesteps to verify that NSDR selections produce feasible solutions. For value tensor evaluations of predicates, we move each robot by step of 0.01m over the defined four directions and check whether it intersects with any obstacles or other robots.

4.1. Data Collection for NSDR Training

We generate training data from simple deadlock scenarios involving 2 to 5 robots, as shown in Fig. 3. When a deadlock is detected, one active robot is selected to switch to passive status. However, in scenarios with 3 to 5 robots, switching a single robot to passive status might be insufficient to resolve the initial deadlock, requiring further evaluation of the predicates, as the situation has changed after one robot is switched to passive status and we need to make new selections upon the new value tensors of the predicates until remaining active robots are moving towards their targets.

We simulate all possible selections of robots switching from active to passive status when a deadlock is detected and record the arrival time $t^i_{arrival}$ for each robot after all robots have arrived at their assigned targets. Then we link the value tensors of predicates, $P_t^{\mathcal{M}}$, with the optimal selection,



(*a*) Num robots = 2 (*b*) Num robots = 3 (*c*) Num robots = 4 (*d*) Num robots = 5

Figure 3: Training scenarios for NSDR, in which a team of robots need to pass through the corridor formed by two rectangular obstacles and arrive at the target position (shown as the red point).

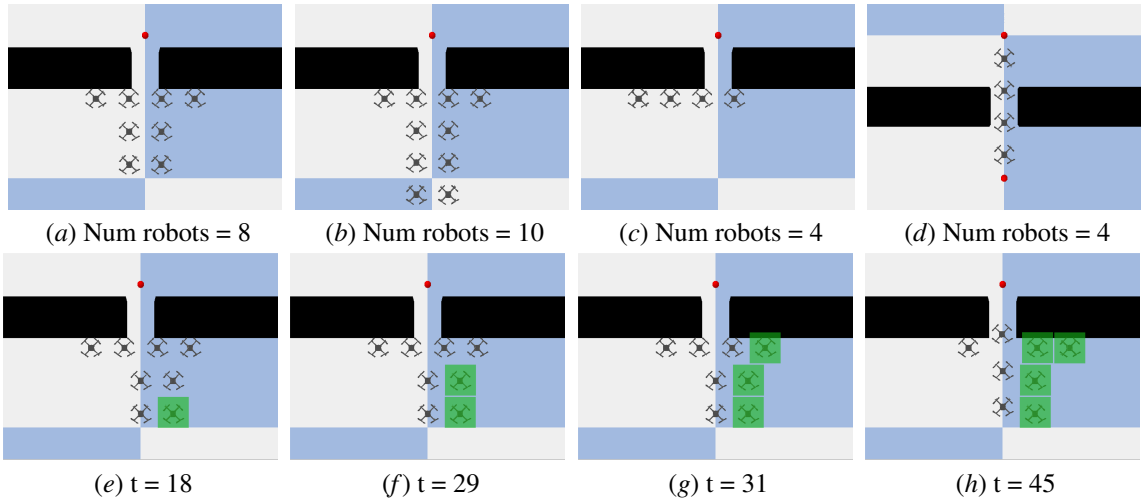
$i^{P_t^{\mathcal{M}}}_{optimal}$, that minimizes $J = \sum_i^M \rho^i t^i_{arrival}$ when select robot with index $i^{P_t^{\mathcal{M}}}_{optimal}$ to switch to the passive status when a deadlock is detected at timestep t and the value tensor of predicates is $P_t^{\mathcal{M}}$. For data augmentation, we randomly assign priorities of 1 or 2 to each robot across 10 runs. From simulations involving 2 to 5 robots, we gathered 8,112 pairs of $P_t^{\mathcal{M}}$ and $i^{P_t^{\mathcal{M}}}_{optimal}$. We then trained our NLM model with depth $D = 3$ and $K = 8$ unary latent conclusions using supervised learning with the value tensor to be the input, and the optimal selection to be the desired output. After training over 25 epochs, we achieved a validation accuracy of 99.01% using 5-folds cross-validations.

4.2. Comparison Analysis

We compare our proposed methods with the adaptive repulsive force (ARF) method from Chen et al. (2023). The ARF method introduces an additional penalty term, $\lambda_{i,j}\|\gamma - w_{i,j}\|_2^2$, into objective function defined in (2). Here, $\lambda_{i,j}$ is a penalty coefficient reflecting the importance of robot m^i relative to robot m^j, γ is a design parameter to maintain a distance γ between robots, and $w_{i,j}$ is the minimum distance between the interiors of robots $\mathcal{P}^i_{rob}$ and $\mathcal{P}^j_{rob}$. The penalty coefficient $\lambda_{i,j}$ is set based on an importance metric ϕ_i assigned to each robot m^i using a greedy approach. When a deadlock is detected, the robot closest to its target receives the highest priority, ϕ_{max}, while the others receive ϕ_{min}. Consequently, $\lambda_{i,j} = \lambda_{max}$ if $\phi_i > \phi_j$, prioritizing the robot with ϕ_{max} to push others aside and reach its target. After the highest-priority robot arrives, importance is reassigned to select the next robot to proceed in a similar manner.

We divide our simulation results into two groups: in-sample scenarios that the NSDR method has trained on and out-of-sample scenarios for testing NSDR's generalization. Note that Greedy and ARF ignore robot priorities and thus behave identically regardless of ρ^i. In contrast, NSDR can minimize the weighted cost $J = \sum_{i=1}^M \rho^i t^i_{\text{arrival}}$, allowing higher-priority robots to reach their goals sooner, which is beneficial in mission-critical tasks. To highlight the consideration of prioritization, Table 1 shows two sets of results: weighted results, averaged over 10 runs with randomly assigned priorities of 1 or 2, and unweighted results, where every robot has a priority of 1.

For in-sample scenarios, we observe that the Greedy method performs slightly better than the ARF method, as resolving deadlocks with only a few robots is relatively straightforward without the need of separating robots away from each other. The ARF method tends to separate robots before resolving deadlocks, whereas the Greedy method's active-passive paradigm keeps passive robots closer to their targets. The NSDR method, however, significantly outperforms both by allowing multiple robots to move toward their targets concurrently, unlike only one robot moves to its assigned targets typically seen in Greedy and ARF. For out-of-sample tests, we first increase the number of robots in deadlocks. NSDR generalizes well, applying its learned logic from sim-



(*a*) Num robots = 8 (*b*) Num robots = 10 (*c*) Num robots = 4 (*d*) Num robots = 4

(*e*) t = 18 (*f*) t = 29 (*g*) t = 31 (*h*) t = 45

Figure 4: Test scenarios for NSDR.

Scenario \ Method		Greedy		NSDR		ARF	
		Weighted	Unweighted	Weighted	Unweighted	Weighted	Unweighted
In Sample	Fig. 3(*c*)	718.7	513	625.8	476	794.2	556
	Fig. 3(*b*)	1105.8	790	785.1	606	1117.3	829
Out of Sample	Fig. 4(*a*)	3769.9	2615	3275.4	2203	3404.4	2337
	Fig. 4(*b*)	4838.7	3339	3505.4	2583	4288.9	2933
	Fig. 4(*c*)	733.6	549	542.5	405	724.9	534
	Fig. 4(*d*)	984.7	718	726.9	581	Deadlock	Deadlock

Table 1: Comparison of the proposed methods across scenarios shown in Fig. 3 and Fig. 4. Weighted objective defined as $J = \sum_i^M \rho^i t^i_{arrival}$ and unweighted objective defined as $J = \sum_i^M t^i_{arrival}$, where ρ_i is the assigned priority and $t^i_{arrival}$ the arrival time of robot i.

ple deadlocks to resolve more complex ones effectively, as shown in Fig. 4(*a*) and Fig. 4(*b*), and it shows a significant performance compared with the other two method. Additionally, it handles completely new scenarios: Fig. 4(*c*) depicts an asymmetrical deadlock, and Fig. 4(*d*) shows two groups of robots aiming for distinct target regions as the top two robots trying to get to the red target position on the bottom and vice versa. In both cases, NSDR demonstrates superior performance compared to the other methods. Notably, the ARF method often creates new force equilibrium, thus new persistent deadlocks, after assigning one robot the highest priority, as in the example of Fig. 4(*d*). We show an exemplary deadlock resolution using NSDR with green shaded robots being selected passive from Fig. 4(*e*) to Fig. 4(*h*) with increasing timesteps for a multi-robot system of 8 robots with the initial deadlock position shown in Fig. 4(*a*).

5. Conclusion

We developed the NSDR method to resolve deadlocks in multi-robot motion planning post-occurrence. After training NSDR on simple scenarios with few robots, we tested it on more complex scenarios with different number of robots. NSDR outperforms the Greedy method and the Adaptive Repulsive Force in resolving deadlocks. Future work includes applying transfer learning to use NSDR trained in simulations for real-world applications, as the learned logic should generalize. Additionally, improving feasibility verification in NSDR is essential for real-time performance.

Acknowledgments

This work is sponsored in part by the ONR under agreement N00014-23-1-2206, AFOSR under the award number FA9550-19-1-0169, and by the NSF under NAIAD Award 2332744 as well as the National AI Institute for Edge Computing Leveraging Next Generation Wireless Networks, Grant CNS-2112562.

References

Mohammed Abdullhak and Andrew Vardy. Deadlock prediction and recovery for distributed collision avoidance with buffered voronoi cells. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 429–436. IEEE, 2021.

Senthil Hariharan Arul, Jong Jin Park, and Dinesh Manocha. Ds-mpepc: Safe and deadlock-avoiding robot navigation in cluttered dynamic scenes. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2256–2263. IEEE, 2023.

Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

Yuda Chen, Chenghan Wang, Meng Guo, and Zhongkui Li. Multi-robot trajectory planning with feasibility guarantee and deadlock resolution: An obstacle-dense environment. *IEEE Robotics and Automation Letters*, 8(4):2197–2204, 2023.

Yuda Chen, Meng Guo, and Zhongkui Li. Deadlock resolution and recursive feasibility in mpc-based multi-robot trajectory generation. *IEEE Transactions on Automatic Control*, 2024.

Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. *arXiv preprint arXiv:1904.11694*, 2019.

Tingxiang Fan, Pinxin Long, Wenxi Liu, and Jia Pan. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*, 39(7):856–892, 2020.

Jaskaran Singh Grover, Changliu Liu, and Katia Sycara. Deadlock analysis and resolution for multi-robot systems. In *Algorithmic Foundations of Robotics XIV: Proceedings of the Fourteenth Workshop on the Algorithmic Foundations of Robotics 14*, pages 294–312. Springer, 2021.

Jaemin Lee, Jeeseop Kim, and Aaron D Ames. Hierarchical relaxation of safety-critical controllers: Mitigating contradictory safety conditions with application to quadruped robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2384–2391. IEEE, 2023.

Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.

Luca Paparusso, Shreyas Kousik, Edward Schmerling, Francesco Braghin, and Marco Pavone. Zapp! zonotope agreement of prediction and planning for continuous-time collision avoidance with discrete-time dynamics. *arXiv preprint arXiv:2406.01814*, 2024.

Jungwon Park, Dabin Kim, Gyeong Chan Kim, Dahyun Oh, and H Jin Kim. Online distributed trajectory planning for quadrotor swarm with feasibility guarantee using linear safe corridor. *IEEE Robotics and Automation Letters*, 7(2):4869–4876, 2022.

Hardik Parwana, Ruiyang Wang, and Dimitra Panagou. Algorithms for finding compatible constraints in receding-horizon control of dynamical systems. In *2024 American Control Conference (ACC)*, pages 2074–2081. IEEE, 2024.

Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. Weighted buffered voronoi cells for distributed semi-cooperative behavior. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 5611–5617. IEEE, 2020.

Rahul Tallamraju, Sujit Rajappa, Michael J Black, Kamalakar Karlapalem, and Aamir Ahmad. Decentralized mpc based obstacle avoidance for multi-robot target tracking scenarios. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–8. IEEE, 2018.

Sarah Tang, Justin Thomas, and Vijay Kumar. Hold or take optimal plan (hoop): A quadratic programming approach to multi-robot trajectory generation. *The International Journal of Robotics Research*, 37(9):1062–1084, 2018.

Charbel Toumieh and Alain Lambert. Decentralized multi-agent planning using model predictive control and time-aware safe corridors. *IEEE Robotics and Automation Letters*, 7(4):11110–11117, 2022.

Chenggang Wang, Shanying Zhu, Bochen Li, Lei Song, and Xinping Guan. Time-varying constraint-driven optimal task execution for multiple autonomous underwater vehicles. *IEEE Robotics and Automation Letters*, 8(2):712–719, 2022.

Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, 2017.

Heqing Yin, Chang Wang, Chao Yan, Xiaojia Xiang, Boliang Cai, and Changyun Wei. Deep reinforcement learning with multi-critic td3 for decentralized multi-robot path planning. *IEEE Transactions on Cognitive and Developmental Systems*, 2024.

Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.