A Co-Evolutionary Multi-Agent Path Planning Framework Using NURBS for Non-Holonomic Robots

Elias J. R. Freitas[1], Miri Weiss Cohen[2], Frederico G. Guimarães[3] and Luciano C. A. Pimenta[4]

***Abstract*—Multi-agent path planning in continuous spaces poses significant challenges due to inter-agent coupling, kinematic constraints, and temporal collision avoidance requirements. This paper proposes a cooperative co-evolutionary planning framework in which each agent evolves its own path and a constant tracking speed while implicitly coordinating with other agents through shared evolutionary information. Robot paths are represented using Non-Uniform Rational B-Splines, enabling smooth and kinematically feasible trajectories for non-holonomic robots. Simulation and real-world experiments demonstrate that the proposed approach reliably generates coordinated, collision-free solutions in complex and cluttered environments, achieving stable convergence within a small number of evolutionary epochs. The results further demonstrate safe and robust path execution under bounded disturbances, ensured by safety margins derived from the employed vector field.**

***Index Terms*—Multi-agent path planning; Cooperative co-evolution; Non-holonomic robots; Kinematic constraints; NURBS.**

I. Introduction

Many real optimization problems in engineering are inherently large-scale and highly coupled, such as Multi-Agent Path Planning (MAPP). The objective is to compute collision-free and coordinated paths that allow multiple agents to move from their initial configurations to designated goals within the same environment [1]. Unlike single-robot planning, MAPP must explicitly account for the interactions among agents, which introduces strong coupling between decision variables and significantly increases the dimensionality of the optimization problem.

Several approaches have been proposed to address the MAPP problem, including centralized and decentralized optimization methods, decoupled planning strategies, and graph- or grid-based planners, as well as evolutionary- and learning-based approaches [2]. Many of these approaches rely on discretized representations of the environment, which often result in poor path quality when applied to continuous spaces. To address these challenges, some methods introduce a smoothing step to improve the quality of the generated paths. For example, Janovská and Surynek [3] combine conflict-based search (CBS) with an RRT* global planner and subsequently smooth the paths using B-spline curves. Even so, particularly when non-holonomic constraints are considered, the paths generated by such planners may become infeasible to execute, as they often contain turns that violate the robot's kinematic limitations.

Incorporating kinematic constraints directly into the planning process further increases the complexity of multi-agent path planning. Recently, Moldagalieva et al. [4] proposed an approach based on CBS that considers kinematic constraints using three levels of operations, where constraints arise directly from conflicts and represent states to be avoided during the search.

In the context of offline planners, our previous work [5] demonstrated that Non-Uniform Rational B-Splines (NURBS) provide a flexible and unified representation capable of generating continuous and differentiable paths with controllable curvature. This idea was later extended to an online planning framework [6], where both the path and the corresponding speed were generated for a single robot. Moreover, in a different context, air corridor planning, a cooperative co-evolutionary approach was proposed to generate multiple paths in three-dimensional space [7]. In this paper, we build upon and unify these approaches by incorporating temporal collision avoidance in multi-agent systems and enhancing robustness to disturbances in the low-level controller speed. The main contributions of this work are summarized as follows:

- a cooperative co-evolutionary framework, combining a planner with a robust curve-tracking controller, for multi-agent path planning in continuous spaces, providing feasible paths and a constant tracking speed for each agent;
- a constrained optimization formulation that combines temporal collision avoidance, obstacle avoidance, and kinematic constraints;
- a NURBS-based path representation with a fully uniform knot vector, enabling smooth and flexible generation of kinematically feasible paths for non-holonomic robots;
- validation through simulations and real-robot experiments, demonstrating robustness under control disturbances.

II. Problem statement

We consider the multi-agent path planning problem in a continuous 2D workspace for N non-holonomic robots. Each robot i must navigate from a given start pose (position

[1]Elias José de Rezende Freitas is with the Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – Campus Ibirité, Ibirité, MG, Brazil. elias.freitas@ifmg.edu.br

[2]Miri W. Cohen is with Braude College of Engineering, Karmiel, Israel.

[1,3,4] Frederico G. Guimarães and Luciano C. A. Pimenta are with Universidade Federal de Minas Gerais, UFMG, Brazil.

and orientation) $[x_i^s, y_i^s, \gamma_i^s]$ to a goal pose $[x_i^e, y_i^e, \gamma_i^e]$ while satisfying the following feasibility requirements:

- **Temporal collision avoidance:** robots must avoid collisions with each other, considering their velocities and timing along the planned path.
- **Obstacle avoidance:** robots must avoid collisions with static obstacles in the environment;
- **Kinematic constraints:** the planned path should respect the kinematic constraints of the robots.

In this work, it is assumed that the kinematic constraint is expressed as a bound on the maximum curvature of the robot. Each robot is modeled as a differential-drive platform with linear speed v_i and angular speed $\omega_i \leq \omega_i^{\max}$. This implies that the maximum admissible curvature is given by:

$$\kappa_i^{\max}(v_i) = \frac{\omega_i^{\max}}{v_i}. \tag{1}$$

Moreover, each robot has a predefined desired speed v_i^{pref} to follow a path. In addition, the robots can adjust their speed within a constrained range $[v_i^{\min}, v_i^{\max}]$. Then, the main objective is to find feasible paths and corresponding linear speeds $\{C_i, v_i\}_{i=1}^N$ that minimize the total travel time of all robots, while accounting for disturbances in the speed reference v_i and maintaining proximity to the preferred speed v_i^{pref}.

III. Proposed methods

The proposed MAPP framework is composed of three main components: (A) path representation based on NURBS curves employing a fully uniform knot vector; (B) solution of the defined constrained optimization problem through a cooperative co-evolutionary approach; and (C) path following with optimized speeds based on a robust vector field. During the optimization process, disturbances affecting each robot's speed are accounted for through a safety margin defined by the selected robust vector field.

A. Path representation

Each robot follows with a constant linear speed v_i a path $C_i(s)$, represented by a k-degree NURBS curve parameterized by $s \in [0, 1]$:

$$C(s) = \frac{\sum_{i=0}^{N_p-1} w_i B_{i,k}(s) \mathbf{P_i}}{\sum_{i=0}^{N_p-1} w_i B_{i,k}(s)} = \sum_{i=0}^{N_p-1} R_{i,k}(s) \mathbf{P_i}, \tag{2}$$

where $R_{i,k}(s) = \frac{w_i B_{i,k}(s)}{\sum_{i=0}^{N_p-1} w_i B_{i,k}(s)}$ are the rational basis functions. A NURBS curve consists of a set of $N_p \geq k+1$ control points: $\mathcal{P} = \{\mathbf{P_0}, \mathbf{P_1}, ..., \mathbf{P_{N_p-1}}\}$. Each weight $w_i \in \mathbb{R}_+^*$ is associated with a control point, which forms the set $\mathbf{w} = \{w_0, w_1, ..., w_{N_p-1}\}$, and $B_{i,k}(s)$ are the normalized B-spline basis functions of $k-$degree, recursively obtained [8] In this study, unlike previous works, we employ a uniform knot vector, as an alternative to the *clamped* uniform knot vector:

$$\mathbf{h} = \frac{1}{N_p + k} [0, 1, 2, \ldots, N_p + k]. \tag{3}$$

To ensure that the path starts and ends at the desired poses using this uniform knot vector, we assign $(k+1)$ repeated control points at both the start and end positions. This configuration can benefit the speed controller, as all derivatives up to order k at the start and end points become zero. Additionally, two extra control points are inserted, aligned collinearly with the start and end points, respectively $\mathbf{P_i^0} = [x_i^s, y_i^s]^T$ and $\mathbf{P_i^{N_p-1}} = [x_i^e, y_i^e]^T$, ensuring the desired orientations:

$$\begin{aligned}
\mathbf{P_i^{k+1}} &= \mathbf{P_i^0} + \frac{2}{3}\lambda \begin{bmatrix} \cos\gamma_i^s \\ \sin\gamma_i^s \end{bmatrix}, \\
\mathbf{P_i^{k+2}} &= \mathbf{P_i^{k+1}} + \frac{1}{3}\lambda \begin{bmatrix} \cos\gamma_i^s \\ \sin\gamma_i^s \end{bmatrix}, \\
\mathbf{P_i^{N_p-k-2}} &= \mathbf{P_i^{N_p-1}} - \frac{2}{3}\lambda \begin{bmatrix} \cos\gamma_i^e \\ \sin\gamma_i^e \end{bmatrix}, \\
\mathbf{P_i^{N_p-k-3}} &= \mathbf{P_i^{N_p-k-2}} - \frac{1}{3}\lambda \begin{bmatrix} \cos\gamma_i^e \\ \sin\gamma_i^e \end{bmatrix}.
\end{aligned} \tag{4}$$

where a constant λ balances the size of the inserted collinear segments. Then, the complete set of control points $\mathcal{P}_i$ that define the curve $C_i(s)$, considering $N_{p_{\text{free}}} = N_p - 2(k+3)$ free control points, is given by:

$$\begin{aligned}
\mathcal{P}_i = [& \underbrace{\mathbf{P_i^0}, \ldots, \mathbf{P_i^0}}_{(\mathbf{k+1})}, \underbrace{\mathbf{P_i^{k+1}}, \mathbf{P_i^{k+2}}}_{\text{collinear points}}, \\
& \underbrace{\mathbf{P_i^{k+3}}, \ldots, \mathbf{P_i^{k+2+N_{p_{\text{free}}}}}}_{\text{free control points}}, \\
& \underbrace{\mathbf{P_i^{N_p-k-2}}, \mathbf{P_i^{N_p-k-1}}}_{\text{collinear points}}, \\
& \underbrace{\mathbf{P_i^{N_p-1}}, \ldots, \mathbf{P_i^{N_p-1}}}_{(\mathbf{k+1})}],
\end{aligned} \tag{5}$$

λ is set to twice the robot radius and assigned unitary weights to the fixed control points $w_i^0 = w_i^1 = \cdots = w_i^k = w_i^{k+1} = w_i^{k+2} = w_i^{N_p-k-3} = w_i^{N_p-k-2} = \cdots = w_i^{N_p-1} = 1$.

B. Cooperative Co-evolutionary approach

The problem described in Section II can be defined as the following constrained optimization problem:

$$\boldsymbol{\chi}^* = \arg\min_{\boldsymbol{\chi}} \left[\alpha_1 \sum_{i=1}^{N} T_i \; + \; \alpha_2 \sum_{i=1}^{N} \left(v_i - v_i^{\text{pref}}\right)^2 \right],$$

s.t.:

$$\begin{aligned}
& C_i(s) \in \Omega_{\text{free}}, \forall i \in [1, N] \\
& \kappa_i(s) \leq \kappa_{\max}(v_i), \forall i \in [1, N] \\
& v_i^{\min} \leq v_i \leq v_i^{\max}, \forall i \in [1, N] \\
& \mathbf{L} \leq \boldsymbol{\chi}_i \leq \mathbf{U}, \forall i \in [1, N] \\
& \|C_i(t) - C_j(t)\| \geq d_{\min}, \quad \forall t \in [0, T_i], \; \forall i \neq j, \\
& \qquad \text{where } s(t) \text{ satisfies } t = \int_0^{s(t)} \frac{\|C_i'(u)\|}{v_i} du,
\end{aligned} \tag{6}$$

where T_i denotes the travel time of agent i, defined as $T_i = \int_0^1 \frac{\|C_i'(s)\|}{v_i} ds$. Assuming a constant linear speed v_i

and applying the arc-length transformation, we obtain the trajectory $C_i(s(t)) \equiv C_i(t)$, which is used to check temporal collisions along the paths. The optimal solution vector $\boldsymbol{\chi}^* = [\boldsymbol{\chi}_0^*, \boldsymbol{\chi}_1^*, \ldots, \boldsymbol{\chi}_N^*]$ contains the solutions for all agent's paths and speeds. The vectors $\mathbf{L} = [L_1, L_2, \ldots, L_D]$ and $\mathbf{U} = [U_1, U_2, \ldots, U_D]$ denote the lower and upper bounds of each decision vector $\boldsymbol{\chi}_i \in \mathbb{R}^D$, where $D = 3N_{p_{\text{free}}} + 1$, accounting for the two-dimensional free control points (displacements), their associated weights, and the path-following speed. Note that the dependence on the candidate solution $\boldsymbol{\chi}$ arises because the path and speed for each agent i are determined from the components of $\boldsymbol{\chi}_i$, detailed below. More explicitly, this dependence can be expressed as $C_i(s) \equiv C_i(s, \boldsymbol{\chi}_i)$ and $v_i \equiv v_i(\boldsymbol{\chi}_i)$. In addition, Ω_{free} represents the free space, accounting for the presence of static obstacles, and $d_{\min}$ denotes the minimum safety distance, ensuring collision-free motion even under disturbances (this parameter is discussed in the next section).

To solve this NP-hard problem, we employ the Cooperative Co-evolutionary LSHADE-COP approach [7], summarized in Algorithm 1. The main idea is to divide the problem into subproblems and co-evolve them using a random ring structure, where the best individuals of each subproblem are shared in each iteration. This shared information is then used to evaluate whether a trajectory is collision-free with respect to other agents while optimizing the path, and also to find the speed to follow it for each agent. The employed solver LSHADE-COP [5] enhances LSHADE [9] by explicitly addressing constraints through a feasibility-aware population management strategy. By guiding mutation and selection using feasibility rules [10, 11], the method is particularly effective for path planning applications that require simultaneous satisfaction of collision avoidance and kinematic constraints [5, 7, 6].

Each subproblem i is reformulated as the following optimization problem:

$$\boldsymbol{\chi}^* = \arg\min_{\boldsymbol{\chi}} \left[\alpha_1 T_i \; + \; \alpha_2 \big(v_i - v_i^{\text{pref}}\big)^2 \right],$$

s.t.:

$$\begin{aligned} h_0(\boldsymbol{\chi}_i) &= \sum_{s \in \mathcal{S}} \max\left(0,\; d_{\min} - d_{\text{obs}}(C_i(s))\right) = 0, \\ h_1(\boldsymbol{\chi}_i) &= \sum_{s \in \mathcal{S}} \max\left(0,\; \kappa_i(s) - \kappa_{\max}\right) = 0, \\ h_2(\boldsymbol{\chi}_i) &= \sum_{t \in [0, T_i]} \max\left(0,\; f_{\text{time}}(C_i(t), \boldsymbol{\chi}_{\text{shared}}, d_{\min})\right) = 0, \end{aligned} \tag{7}$$

where the individual solution is a set of parameters to obtain the path C_i and the speed v_i for each robot, described by

$$\boldsymbol{\chi}_i^* = \begin{bmatrix} \Delta\mathbf{P}_{k+3}, \Delta\mathbf{P}_{k+4}, \ldots, \Delta\mathbf{P}_{k+1+N_{p_{\text{free}}}}, \\ w_{k+3}, w_{k+4}, \ldots, w_{k+1+N_{p_{\text{free}}}}, v_i \end{bmatrix}. \tag{8}$$

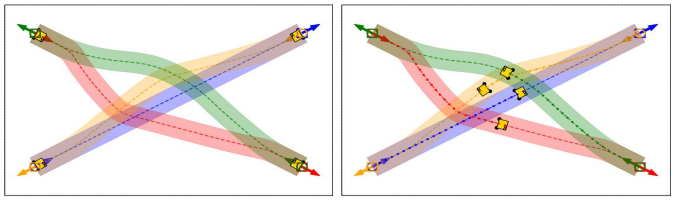
Note that we use the displacements of the NURBS control points relative to the previous curve for each epoch.
Each constraint function is designed to return zero when fully satisfied, and a positive penalty value when violated. The equality constraint $h_0(\boldsymbol{\chi}_i)$ ensures that all discretized samples $s \in \mathcal{S}$ of curve $C_i(s)$ remain within the free space Ω_{free}. This is achieved by enforcing that the distance from each curve point $C_i(s)$ to the nearest obstacle, denoted $d_{\text{obs}}(C_i(s))$, is greater than or equal to the prescribed safety margin $d_{\min}$. Additionally, $h_1(\boldsymbol{\chi}_i)$ imposes a curvature constraint along the entire path, ensuring $\kappa_i(s) \leq \kappa_{\max}$. Finally, $h_2(\boldsymbol{\chi}_i)$ accumulates temporal violations returned by the function f_{time} based on the minimum safety distance $d_{\min}$, the shared information $\boldsymbol{\chi}_{\text{shared}}$ between agents (e.g., the best curve from the previous epoch, $C_j, \forall j \neq i$), thereby enforcing temporal collision avoidance.

Algorithm 1: Co-Evolutionary LSHADE-COP [7]

```
Divide the problem into N subpopulations, each corresponding to an air corridor (agent).
foreach subpopulation χ_0^ℓ (ℓ = 1, 2, ..., N) do
    Initialize the subpopulation χ_0^ℓ ;
    Evaluate χ_0^ℓ ;
    χ_shared^ℓ ← obtain the individual to share with other agents ;
χ_best ← obtain the complete solution of the problem based on the best individuals in the subpopulations ;
f_0, CV_0 ← evaluation of the complete cost function and the sum of constraints for χ_best;
for epoch e = 1 to MaxEpochs do
    S_e ← a randomly shuffled order of agents ;
    foreach subpopulation χ_e^ℓ (ℓ ∈ S_e) do
        χ_best_new^ℓ ← Run LSHADE-COP for χ_e^ℓ using
          χ_shared = [χ_shared^1 ··· χ_shared^N];
        // selection
        χ_best_new ← obtain the complete solution of the problem
          based on the new best individual in the subpopulation i ;
        f_e, CV_e ← evaluate χ_best_new ;
        if ((f_e ≤ f_{e-1}) &
          (CV_e = CV_{e-1} = 0)) or (CV_e ≤ CV_{e-1} &
          CV_{e-1} > 0) then
            χ_shared^ℓ ← update the shared individual;
            χ_best ← χ_best_new ;
            f_{e-1} ← f_e ;
    // Explicit memory approach
    foreach subpopulation χ_e^ℓ (i = 1, 2, ..., n) do
        Re-initialize the subpopulation χ_e^ℓ based on and including
          the best solution ;
        Re-evaluate χ_e^ℓ ;
        χ_shared^ℓ ← obtain the individual to share with other
          agents;
return χ* ← χ_best;
```

C. Vector field-based robust tracking curve

Given the optimized reference path $C_i^*(s)$ for each robot, we employ the path-following approach outlined in [12] to track this path with the optimized speed v_i^*. This approach defines a robust vector field, which is computed online from the current robot position and the given path. For that, we assume that a low-level controller enforces a closed-loop behavior that approximates a single integrator subjected to bounded disturbances when tracking the velocity vector field. This dynamic behavior is represented by $\dot{\mathbf{x}} = \mathbf{u} + \boldsymbol{\delta u}$, where $\mathbf{x}$ denotes the robot's position, $\mathbf{u}$ is the velocity vector



(a) $t = 0$s. (b) $t = 19.2$s.

Fig. 1: Two snapshots of the simulation for Scenario 1 (first case) with four robots, showing the optimized paths (dashed lines), the safety tubular regions, and the robots (in yellow boxes).

reference with $||\mathbf{u}|| = v_i^*$, and $||\boldsymbol{\delta u}|| \leq \Delta u$ accounts for the disturbances.

Overall, Rezende et al.[12] showed that the robot's reference point asymptotically converges to an invariant set $\mathcal{I}$ under dynamic modeling assumptions. This invariant set $\mathcal{I}$ corresponds to a tubular region around the reference curve, with cross-sectional radius r_{control} obtained *a priori* by: $r_{\text{control}} = \frac{1}{k_f} \tan\left(\frac{\pi}{2} \frac{\Delta u}{v_i}\right)$, where k_f is the positive gain associated with the vector field functions. Based on this, and considering a circular robot with radius r_i, the safety margin is defined as $d_{\min} \geq r_i + r_{\text{control}}$ and applied in (7).

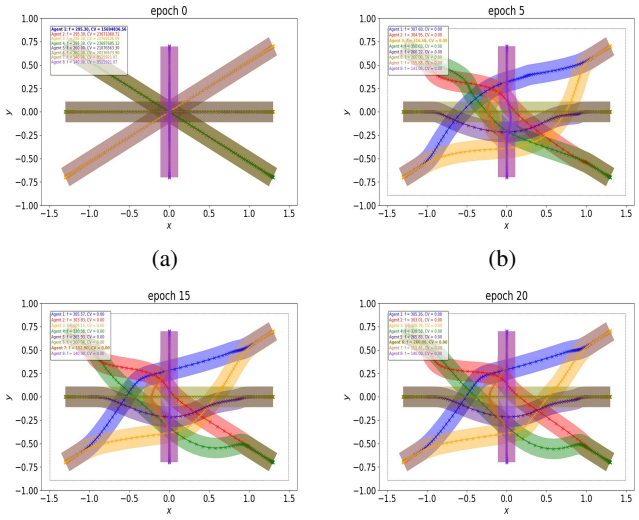
IV. Experimental Setup

The algorithms were implemented in Python using the GEOMDL library [13] to handle NURBS curves. All experiments were conducted on an Intel Core i9-12900KF CPU running at up to 5.1 GHz with the Ubuntu 20.04 Linux operating system. The Robotarium environment [14] was used for both simulated and real experiments, featuring a 3.2m×2.0m workspace. The maximum angular speed of the differential-drive robots was limited to 0.8rad/s, the preferred speed was set to $v_i^{\text{pref}} = 0.1$m/s, and we estimated the safe distance $d_{\min} = 21.5$cm. The NURBS curve has $N_{p_{\text{free}}} = 4$ free control points and was discretized into 150 points. The LSHADE-COP algorithm was configured to stop after 1000 cost function evaluations, with a population size of 117 ($= 9 \cdot D$), $MaxEpochs$ set to 20, and the function weights set to $\alpha_1 = \alpha_2 = 1$. Supplementary materials, including experimental videos, are available on GitHub at https://github.com/eliasjof/multiagent_pathplanning.

V. Results and Discussion

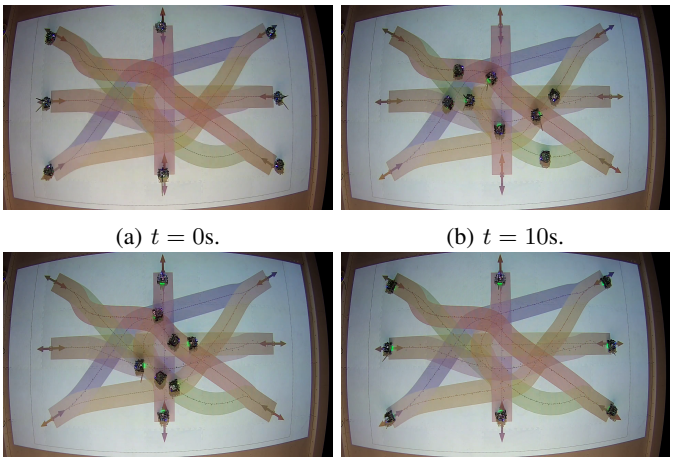
A. Scenario 1: Free-obstacle environment

In the first scenario, we examined two cases involving four and eight robots navigating an obstacle-free environment. Each robot is tasked with reaching the position directly opposite to its initial location while maintaining the same orientation as at the start. Figure 1 shows snapshots of the simulation performed in the Robotarium environment, where the employed vector field was used to track the respective paths at a constant linear speed. The optimal linear speed v_i obtained for all the robots was 0.1m/s.



(a) (b) (c) (d)

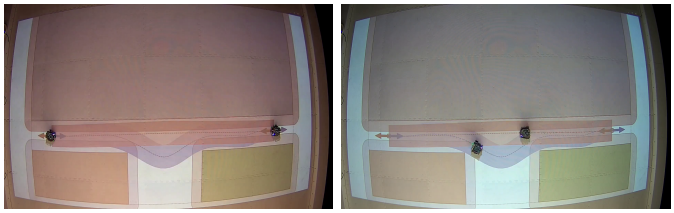
Fig. 2: Evolution of the set of paths generated in Scenario 1 (second case) with eight robots. Each path (dashed lines) is enclosed by a tubular region representing the safety margin.



(a) $t = 0$s. (b) $t = 10$s. (c) $t = 16$s. (d) $t = 33$s.

Fig. 3: Snapshots of the real-world experiment conducted in the Robotarium (top-view camera) using eight robots for Scenario 1 (second case). The paths (dashed lines) and the safety tubular region are projected onto the environment for visualization purposes only.

Figure 2 presents the evolution of the paths generated over the epochs for the second case (eight robots in the environment), revealing also a deviation pattern formed through the co-evolutionary optimization process. Snapshots of the real-world experiments are shown in Figure 3. Despite errors in the commanded velocities during the real-world experiments, the robots successfully avoided collisions while maintaining a predefined safety margin represented by the tubular regions. The optimized tracking velocities for each agent were $\mathbf{v} = [0.100, 0.100, 0.100, 0.093, 0.100, 0.100, 0.100, 0.100]$ m/s.



(a) $t = 0$s. (b) $t = 10$s.

(c) $t = 12$s. (d) $t = 24.5$s.

Fig. 4: Snapshots of the real-world experiment conducted in the Robotarium (top view camera) using two robots for Scenario 2 in a narrow passage. The paths (dashed lines), the corresponding safety tubular regions, and the colored obstacles are projected onto the environment for visualization purposes only.

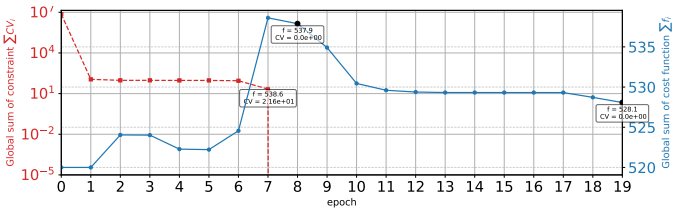
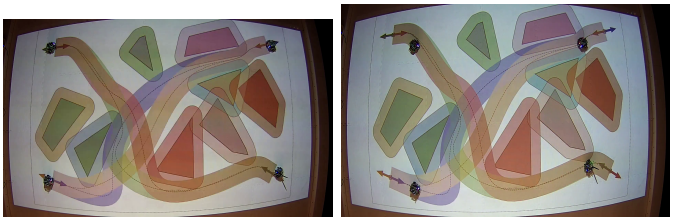


Fig. 5: Evolution of the global sum of the cost function and the total constraint violation over the optimization epochs for Scenario 2.

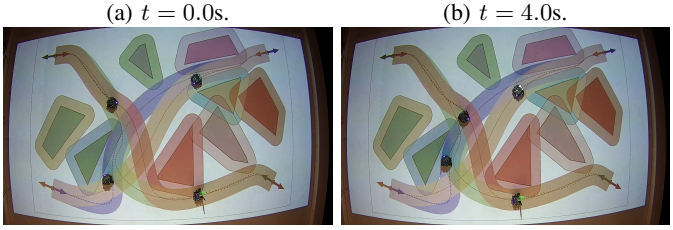
B. Scenario 2: Narrow passage

In the second scenario, two robots must reach opposite sides through a narrow corridor, where only one robot can pass at a time. A small escape area is available, allowing one robot to temporarily move aside and let the other pass. This configuration evaluates the planner's capability to generate an optimal coordination strategy that minimizes the overall mission time. Snapshots of the real-world experiment conducted in the Robotarium are shown in Figure 4. The robots successfully executed the planned trajectories while maintaining safety margins and avoiding collisions in the narrow passage.

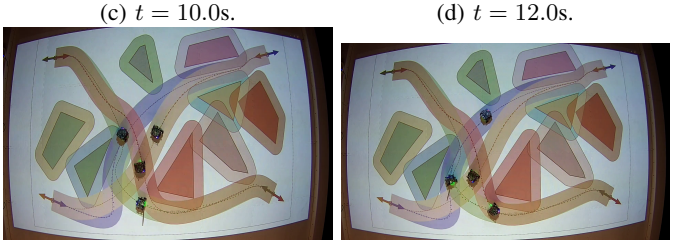
As shown in Figure 5, after seven epochs, the solution becomes feasible, exhibiting consistent convergence behavior. It is worth noting that, in LSHADE-COP, feasible solutions are always preferred over infeasible ones during the selection process, even if they present worse objective function values. Once feasibility is achieved, the algorithm focuses on minimizing the objective function. The NURBS-based path representation allows the robots to maintain the maximum linear speed of 0.1 m/s, deviating only when necessary to ensure collision-free motion and to minimize the overall mission time.



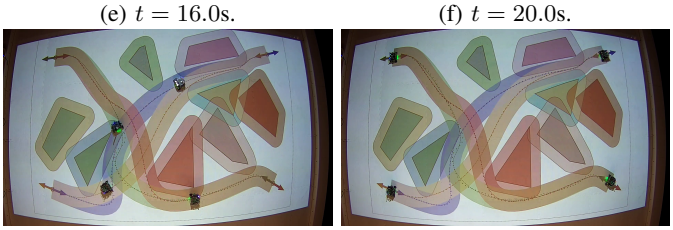
(a) $t = 0.0$s. (b) $t = 4.0$s.



(c) $t = 10.0$s. (d) $t = 12.0$s.



(e) $t = 16.0$s. (f) $t = 20.0$s.



(g) $t = 24.0$s. (h) $t = 36.3$s.

Fig. 6: Snapshots of the real-world experiment conducted in the Robotarium (top view camera) using four robots for Scenario 3. The paths (dashed lines), the corresponding safety tubular regions, and the colored obstacles are projected onto the environment for visualization purposes only.

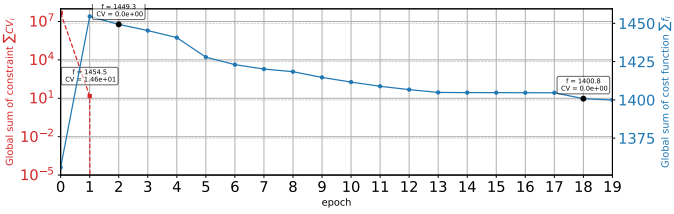


Fig. 7: Evolution of the global sum of the cost function and the total constraint violation over the optimization epochs for Scenario 3.

C. Scenario 3: Cluttered environment

In the third scenario, four robots must reach their respective goals in an environment containing eight obstacles, represented as randomly placed and shaped polygons. In one section of the workspace, the obstacles form a narrow passage that

allows only one robot to pass at a time, thereby making the optimization problem more challenging.

Snapshots from the real-world experiment conducted in the Robotarium are presented in Figure 6. The robots successfully executed the optimized trajectories, maintaining their safety margins and effectively coordinating to traverse the narrow passages. These experimental results validate the proposed method in a real-world, cluttered multi-robot environment. Although some sections of the paths intersect, the convergence plot in Figure 7 shows a consistent reduction in both the global cost and constraint violation, with the solution becoming feasible after only a few epochs.

VI. Conclusion

This paper presented a co-evolutionary planner framework to address the multi-agent path planning problem. In this formulation, each agent evolves its own path and linear speed while implicitly coordinating with the others through shared information during the evolution process.

The results demonstrated that the co-evolutionary approach effectively handles complex and cluttered environments, where agents must coordinate their movements to traverse narrow passages and avoid mutual temporal collisions. The optimization process exhibited stable convergence behavior, with feasible solutions being reached after only a few epochs. The resulting paths were smooth and kinematically feasible, enabling robust navigation for all agents.

Furthermore, real-world experiments conducted in the Robotarium laboratory validated the applicability of the proposed framework beyond simulation, considering bounded disturbances in the low-level speed controller. This approach is offline and centralized. Another limitation is the high execution time (on the order of several tens of minutes), since it is currently implemented in a naive way in Python, which requires code refactoring for improved efficiency. Extensions of this approach will consider scenarios involving dynamic environments characterized by unpredictable obstacles and target motion. Additionally, incorporating communication constraints within distributed optimization frameworks presents a promising research direction.

References

[1] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, no. 1, 2019, pp. 151–158.

[2] S. Lin, A. Liu, J. Wang, and X. Kong, "A review of path-planning approaches for multiple mobile robots," *Machines*, vol. 10, no. 9, p. 773, 2022.

[3] K. Janovská and P. Surynek, "Multi-agent path finding in continuous environment," in *2024 IEEE 36th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2024, pp. 708–713.

[4] A. Moldagalieva, J. Ortiz-Haro, M. Toussaint, and W. Hönig, "db-cbs: Discontinuity-bounded conflict-based search for multi-robot kinodynamic motion planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 569–14 575.

[5] E. J. R. Freitas, M. W. Cohen, A. A. Neto, F. G. Guimarães, and L. C. A. Pimenta, "DE3D-NURBS: A differential evolution based 3D path-planner integrating kinematic constraints and obstacle avoidance," *Knowledge-Based Systems*, vol. 300, p. 112084, 2024.

[6] E. J. R. Freitas, M. W. Cohen, F. G. Guimarães, and L. C. A. Pimenta, "Deliberative control-aware motion planning for kinematic-constrained UAVs in a dynamic environment," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1. Atlanta, GA, United States: IEEE, 2025, pp. 1–8.

[7] ——, "Air corridor planning for uavs using a cooperative co-evolutionary approach and nurbs representation," in *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*. Charlotte, NC, United States: IEEE, 2025, pp. 518–525.

[8] L. Piegl and W. Tiller, *The NURBS book*, 2nd ed. Berlin: Springer-Verlag Berlin Heidelberg, 1997.

[9] R. Tanabe and A. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE congress on evolutionary computation (CEC)*. Beijing, China: IEEE, 2014, pp. 1658–1665.

[10] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[11] X. Yu, C. Li, and J. Zhou, "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios," *Knowledge-Based Systems*, vol. 204, p. 106209, 2020.

[12] A. M. Rezende, V. M. Gonçalves, A. H. Nunes, and L. C. Pimenta, "Robust quadcopter control with artificial vector fields," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, 2020, pp. 6381–6387.

[13] O. R. Bingol and A. Krishnamurthy, "NURBS-Python: An open-source object-oriented NURBS modeling framework in Python," *SoftwareX*, vol. 9, pp. 85–94, 2019.

[14] S. Wilson, P. Glotfelter, S. Mayya, G. Notomista, Y. Emam, X. Cai, and M. Egerstedt, "The robotarium: Automation of a remotely accessible, multi-robot testbed," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2922–2929, 2021.