ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **27**, Number 1, 2024, 21–36

Shortest Path Planning and Efficient Fuzzy Logic Control of Mobile Robots in Indoor Static and Dynamic Environments

Abdelfetah HENTOUT[1,*], Abderraouf MAOUDJ[2], and Ahmed KOUIDER[1]

[1]Division of Robotics and Industrial Automation (DPR), Centre for Development of Advanced Technologies (CDTA), Baba Hassen, 16303, Algeria
[2]SDU Biorobotics, MMMI, University of Southern Denmark (SDU), Denmark
Email: ahentout@cdta.dz*, abma@mmmi.sdu.dk, akouider@cdta.dz
* Corresponding author

Abstract. Efficient navigation in dynamic environments is a critical skill for of mobile robots, where obstacles can stochastically appear. This paper presents a complete navigation and control system that integrates effective path optimization and motion control capabilities for mobile robots evolving in indoor static and dynamic environments. This system consists primarily of two layers. In the *Optimization Layer* (*Global planner*), a *Deterministic Constructive Algorithm* (*DCA*) quickly generates the shortest path, as a sequence of nodes, to get to the goal position while avoiding the static obstacles. The *Control layer* (*Local planner*) employs an *Efficient Fuzzy Logic Controller* (*EFLC*) to continuously guide the robot around the detected dynamic obstacles and drive safely the robot along the intended path. Simulations conducted on various maps with different complexities demonstrate the effectiveness of the *DCA* planner. Finally, validations using *V-REP* software show the strength of the proposed *EFLC* that mimics human reasoning for mobile robots navigating in dynamic environments.

Key-words: Deterministic constructive algorithm; indoor static and dynamic environments; fuzzy logic control; mobile robots; shortest path planning.

1. Introduction

The ability to autonomously navigate while avoiding static and dynamic obstacles in indoor workspaces (W-spaces) is an important challenge for mobile robots to safely achieve their tasks. Additionally, path optimization is one of the key research issues; further, it is easier in static than dynamic environments. Dealing with such an open problem, several studies have been achieved resulting in a considerable number of approaches that differ according to their performance criteria and type of navigation W-spaces (static or dynamic). Generally, this problem can be tackled by two types of methods: (*i*) *Deterministic approaches* and (*ii*) *Non-deterministic approaches*.

The first category, *Deterministic approaches*, encompasses graph-based algorithms such as Rapidly-exploring Random Tree (RRT), Probabilistic RoadMap (PRM) and A*. Zuo et al. [1] develop a hierarchical path planning approach using A* to quickly find a geometric path. Several way-points are chosen as sub-goals for the next phase. A least-square policy iteration is then employed as an optimization mechanism to identify the optimal sub-objective positions within a predefined radius. Although results were significantly better than those returned with the RRT, it is subject to the constraint that the optimization is only performed on a set of pre-selected points within a limited radius [2]. An enhanced RRT that incorporates BackTracking (BT) and A* is proposed in [3]. Each iteration, RRT-A*-BT selects the optimal path connecting the current position to $Target$. Consequently, the shortest path is determined with a trade-off of a lower convergence rate. Authors in [4] propose the PRM* and RRT* using best-neighbor search procedure and tree rewiring. Both techniques generate shorter paths under low convergence and high computation time. *Deterministic approaches* are generally efficient and cost-effective in small environments. However, their susceptibility to getting stuck in local minima makes them less suitable for complex W-spaces [5]. Moreover, even after exploring the environment and constructing a graph, additional processing is needed to compute a path based on the graph [6]. Owing to their inherent NP-hardness, *deterministic approaches* often struggle to provide satisfactory solutions.

Employing *Non-deterministic approaches* such as metaheuristics becomes essential for escaping local minima and achieving satisfactory results. They can be classified in two main classes: (*i*) *Algorithms using a neighborhood search* that start from an initial solution and apply an improvement procedure by examining neighboring solutions and (*ii*) *Algorithms using a global search* that generate new solutions randomly to progressively enhance the current solutions set. Researchers develop numerous metaheuristic algorithms based on Genetic Algorithms (GA) [7–10], Artificial Bee Colony (ABC) [11,12], Particle Swarm Optimization (PSO) [13–15], Cuckoo Search (CS) [16], Gravitational Search Algorithm (GSA) [17,18], Ant Colony Optimization (ACO) [19, 20, 26], Grey Wolf Optimization (GWO) [21, 22] and Slime Mould Algorithm (SMA) [23]. These methods are significant as they establish reference inputs for tracking problems specific to mobile robots [24, 25].

Within the *Global search-based category*, GA and PSO are widely used. Authors in [10] propose a GA-based technique that employs an encoding scheme to increase the chromosomes flexibility and a new crossover operator for more effectiveness. Li and Chou [14] develop a self-adaptive learning PSO with various strategies for multi-objective optimization, and develop a new self-adaptive learning mechanism to improve its search capacity. Bakdi et al. [9] exploit depth information from a vision system to model the robot W-space. GA technique is then used to construct an optimal collision-free path. Authors in [20] propose an ACO algorithm to balance fast convergence with efficient exploration. This approach uses a non-uniform pheromone distribution to minimize the search space area explored by ACO and improve the time efficiency. Despite their effectiveness, these methods are still limited regarding premature convergence.

To overcome some of these issues, many studies combine *deterministic* and *non-deterministic algorithms*, thus capitalizing on their respective advantages. For instance, authors in [28] combine ACO and Dijkstra algorithms. In this method, ACO refines the initial near-optimal path produced by Dijkstra to construct the final optimal path. Kala et al. [29] develop another approach that hybridizes A* with Fuzzy Logic (FL). Dai and colleagues [19] propose a recent ACO-A* hybrid algorithm, where A* is employed to enhance the convergence speed of ACO.

Other types of approaches such as Machine Learning (ML) and Reinforcement Learning (RL) are proposed for path planning [27]. Authors in [30] develop an Efficient Q-Learning (QL) algorithm to generate optimized courses. Low et al. [31] present an improved QL approach and use a Flower Pollination Algorithm (FPA) to initialize the Q-table. Chang et al. [32] suggest an improved Dynamic Window Approach (DWA) based on QL. DWA is employed for local path planning, while QL is specifically used to adaptively tune its parameters. Even the combination improves performance, it requires high runtime because of the training efforts for large instances [30]. Despite receiving considerable attention, these algorithms are often inefficient because they get trapped in local minima [31]. Another weakness includes the heavy computation requirements to generate near-optimal paths, high operational complexity and implementation difficulties with dynamic obstacles (real-time parameters setting). Thereby, research in this area is still emerging, as there is no unified algorithm integrating all features and providing best solutions in static and dynamic W-spaces. For many applications, the environment changes in such a manner it is impossible to be foreseen by the designer. Finally, information is generally inaccurate and may be incomplete, which may leads to restricted sensor activity.

Addressing several complex issues, particularly avoiding dynamic obstacles, requires an intelligent process such as Artificial Intelligence (AI). For instance, Neural Networks (NN) and FL approaches achieve great success and are suggested as alternatives to conventional methods. Kamil et al. [33] propose an online sensor-based motion planning algorithm using a multi-layer decision FL Controller (FLC) to improve safety and cost. This FLC employs the prediction and priority rules of multi-layer approach for an efficient method. Yuan et al. [34] present a Gated Recurrent Unit-Recurrent NN (GRU-RNN) model for dynamic planning. Inputs and tags are derived from sample sets generated by an improved Artificial Potential Field (APF) and ACO algorithms. An Adaptive Neuro-Fuzzy Inference System (ANFIS) is proposed to construct and optimize the FLC using a set of input/output variables. Juang and Chang [35] develop an FLC based on the evolutionary-group-based PSO. Roman et al. [36] apply the Iterative Feedback Adjustment (IFT) algorithm to solve optimization problems and improve the training speed of NN. Algharbi et al. [37] present an FLC for navigation in unstructured environments, hybridized with other Soft Computing (SC) techniques such as GA, NN, and PSO. SC techniques are used as alternatives to Evolutionary Algorithms (EA) in dynamic environments. Such approaches are ineffective because of getting stuck in local minima and limit cycles. In addition, the final paths accuracy highly depends on the training state (for NN) or conceived rules (for FL) [38]. Finally, such approaches have other implementation issues related to the dynamic nature of the W-spaces, primarily due to the difficulty in real-time parameter adjustment [30].

In summary, the previous categories of path optimization and control approaches face various challenges and often suffer from high design complexity, heavy computation, and implementation challenges, particularly in dynamic W-spaces. *Deterministic methods* are prone to getting stuck in local minima, while *non-deterministic approaches* often struggle with premature convergence. AI-based techniques depend highly on the training states and conceived rules, and encounter difficulties in real-time parameter adjustment.

The main contributions of this paper lie in the development of a complete navigation and control system for mobile robots operating in indoor static and dynamic environments. This is significant because the current literature typically focuses on one aspect, rather than addressing both, simultaneously. The system seamlessly combines two essential capabilities: (*i*) Path planning and optimization, achieved through a *Deterministic Constructive Algorithm* (*DCA*) serving as a global planner, and (*ii*) Motion control, facilitated by an *Efficient Fuzzy Logic Controller* (*EFLC*) acting as a local planner. *DCA* effectively generates the best path for the robot to get to the goal position, addressing the path optimization aspects. Meanwhile, *EFLC* deals with any change in the environment during the robot movement to drive it safely along the intended path, thus addressing the motion control aspect. This combination of capabilities ensures enhanced performance and adaptability for mobile robots in various real-life applications.

The outline of the paper is as follows. Section 2 describes the mathematical model of the problem. Section 3 presents the proposed *Deterministic Constructive Navigation and Path Optimization Algorithm* while describing the *Efficient Fuzzy Logic Controller* for dynamic obstacles avoidance. Section 4 presents and discusses the main obtained results. Finally, Section 5 concludes the paper and presents future work.

2. Formulation of the Problem

The obstacles in an indoor W-space can be effectively approximated using rectangles serving as a suitable representation [30]. A mobile robot is supposed to move from the initial position $Source(x_S, y_S)$ to the final position $Target(x_T, y_T)$. The goal is being to find the shortest path linking $Source$ and $Target$, which consists of a set of n nodes and $(n-1)$ segments. Each segment $Path\_Seg_i(N_i, N_{i+1})$ is given by two successive nodes N_i and $N_{i+1}, i = 1 \ldots n$. The robot evolves in a W-space cluttered with m static obstacles; each is approached by a rectangle of four points $P_1(x_1, y_1) \ldots P_4(x_4, y_4)$ defining four segments: $Obs\_Seg_{tj}, t \in \{1 \ldots 4\}, j \in \{1 \ldots m\}$. The robot is approximated by a rectangle defined by four points; their coordinates vary depending on the robot location. The mathematical equation for each segment, forming a given rectangle wrapping an obstacle, is determined by

$$Seg(P_k, P_l) = \begin{cases} y - y_k = \frac{y_l - y_k}{x_l - x_k}(x - x_k) \\ MIN(x_l, x_k) \leq x \leq MAX(x_l, x_k) \end{cases} \tag{1}$$

Considering the W-space approximation, the path problem consists of finding a set of feasible nodes that optimizes the total length while satisfying condition

$$Path\_Seg_i(P_i, P_{i+1}) \cap Obs\_Seg_{tj} = \phi,\ i = 1 \ldots n, t = 1 \ldots 4, j = 1 \ldots m \tag{2}$$

$$A_{r,tj} = \begin{cases} 1 & if \exists P_1(x_1, y_1), P_2(x_2, y_2) | (\{P_1, P_2\} \in Rob\_Seg_r) \\ & \wedge (\{P_1, P_2\} \in Obs\_Seg_{tj}), \forall t, r \in \{1 \ldots 4\}, \forall j \in \{1 \ldots m\} \\ 0 & otherwise \end{cases} \tag{3}$$

$$B_{i,tj} = \begin{cases} 1 & if \exists P_1(x_1, y_1), P_2(x_2, y_2) | (\{P_1, P_2\} \in Path\_Seg_i) \wedge (\{P_1, P_2\} \\ & \in Obs\_Seg_{tj}), \forall i \in \{1 \ldots n\}, \forall t \in \{1 \ldots 4\}, \forall j \in \{1 \ldots m\} \\ 0 & otherwise \end{cases} \tag{4}$$

The following optimization problem, whose variables will be specified as follows, minimizes the total length of the path:

$$min(\sum_{i=1}^{i=n-1} \|N_{i+1} - N_i\|) \tag{5}$$

Table 1. Notations, parameters, and indices [40]

n	Number of nodes.
m	Number of obstacles.
i	Index of segment formed by i^{th} and $(i+1)^{th}$ nodes; $i \in \{1 \ldots n\}$.
j	Index of obstacle j; $j \in \{1 \ldots m\}$.
k	Indices of points $P_l, l \in \{1 \ldots 4\}$ that define an obstacle k.
r	Index of segment r which defines the robot rectangle; $r \in \{1 \ldots 4\}$.
t	Index of segment t defining the rectangle enveloping an obstacle; $t \in \{1 \ldots 4\}$.
$N_i(x_i, y_i)$	i^{th} node of the path.
Obs_j	j^{th} obstacle; $j \in \{1 \ldots m\}$.
$Path\_Seg_i(N_i, N_{i+1})$	i^{th} segment specified by nodes (N_i, N_{i+1}); $i \in \{1 \ldots n-1\}$.
$Obs\_Seg_{tj}(P_k, P_l)$	t^{th} segment of j^{th} obstacle determined by the points (P_k, P_l); $t, k, l \in \{1 \ldots 4\}, j \in \{1 \ldots m\}$.
$CPos$	The current location of the mobile robot.
$Rob\_Seg_r(P_k, P_l)$	r^{th} segment of the rectangle approximating the robot; $\{P_k, P_l\}$ is calculated according to $CPos$; $r, k, l \in \{1 \ldots 4\}$.

The *Euclidean distance* between nodes N_i and N_{i+1} is given by

$$\|N_{i+1} - N_i\| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \forall i \in \{1 \ldots n-1\} \tag{6}$$

The optimization problem is subjected to the following constraints:

$$(x_{i+1} \neq x_i) \vee (y_{i+1} \neq y_i), \forall i \in \{1 \ldots n-1\} \tag{7}$$

$$\sum_{i=1}^{i=n-1} \sum_{j=1}^{j=m} B_{i,tj} = 0, \;\; \forall i \in \{1 \ldots n-1\}, \forall t \in \{1 \ldots 4\}, \forall j \in \{1 \ldots m\} \tag{8}$$

$$\sum_{j=1}^{j=m} A_{r,tj} = 0, \forall r, t \in \{1 \ldots 4\}, \forall j \in \{1 \ldots m\} \tag{9}$$

$$A_{r,tj} \in \{0,1\}, B_{i,tj} \in \{0,1\}, \forall i \in \{1 \ldots n\}, \forall j \in \{1 \ldots m\} \tag{10}$$

The constraints (7) impose all the generated nodes to be different. The constraints (8) guarantee that the path segments will avoid colliding with obstacles. The constraints (9) ensure that the robot does not encounter obstacles when following the nodes. Finally, the constraints (10) indicate that the variables A and B are binary.

3. Description of the Proposed Approach

The purpose of the navigation and control system is to guide the robot from $Source$ to $Target$ on a map. The diagram given in Fig. 1 describes the process in the navigation and control system:

- **Optimization and path planning (Global offline planner)**: The navigation process starts with the path planner, determining the best path for the robot to get from its $CPos$ to $Target$. To this end, a *Deterministic Constructive Algorithm* (*DCA*) rapidly generates the shortest collision-free path while avoiding static obstacles.
- **Motion control (Local online planner)**: While the robot is following the planned path, the control layer continuously guides the robot around the dynamic obstacles that have not been previously included on the map. For this purpose, an *Efficient Fuzzy Logic Controller* (*EFLC*) deals with any change in the environment during the robot movement while calculating the right and left speeds required to drive safely the robot.

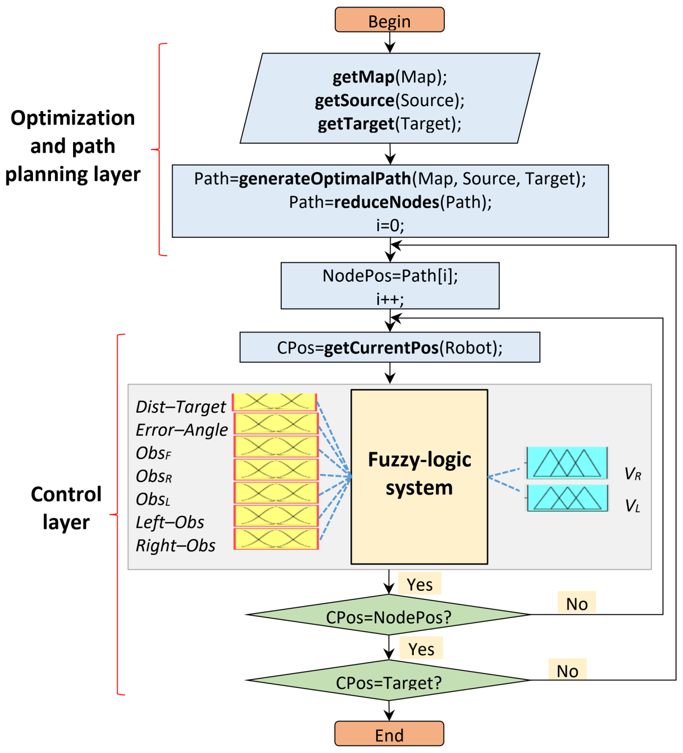


Fig. 1. Overall diagram of the proposed *DCA-EFLC* approach

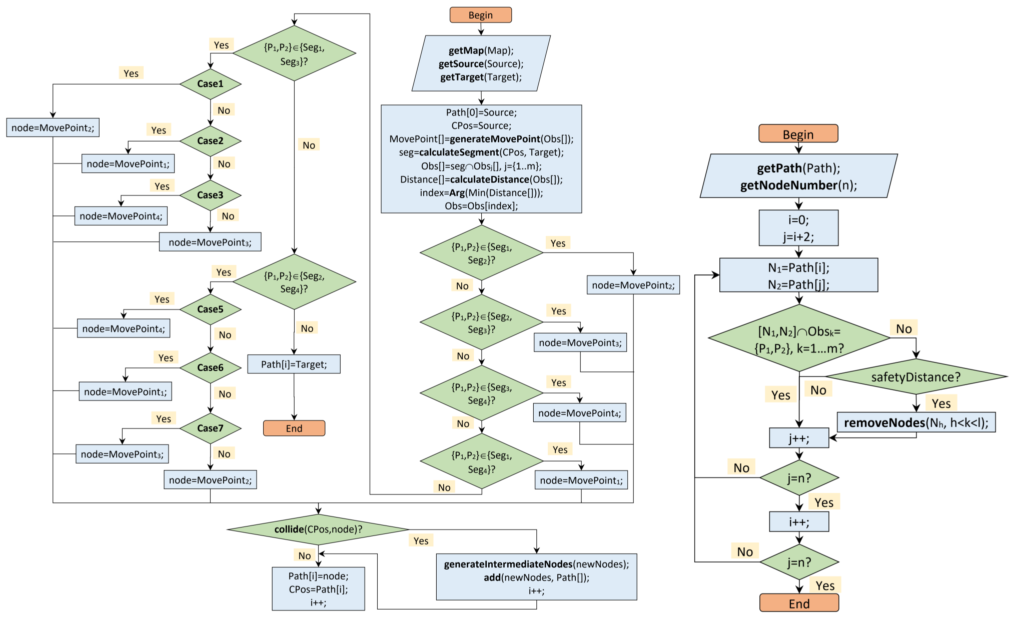
3.1. Collision-free path optimization

The optimization layer is built upon a constructive algorithm [40] to generate the shortest path in two stages: (*i*) *Optimization step* builds the path as an ordered set of nodes; (*ii*) *Reduction of nodes number step* which is a refinement where vain nodes are removed to streamline the path.

The *DCA* incorporates two concepts derived from [40]: (*i*) $MovePoint$ situated at the obstacle corner to enable the robot to maneuver safely around it; and (*ii*) $SafetyDist$ used as a safety distance to define the coordinates $MovePoint_i(x_i, y_i), i = 1 \ldots 4$. This ensures that the robot does not collide with the obstacle while navigating around it.

3.1.1. Optimization algorithm

In Euclidean geometry, the shortest path between two points is the line segment connecting them. When applied to free W-spaces, the straight line is the shortest path linking $Source$ to $Target$. However, in cluttered W-spaces, the obstacles prevent both positions to be connected by a straight line. The shortest path is then built by circumventing obstacles via defined points (nodes) nearly orthogonal to the straight-line segment. Fig. 2a illustrates the overall diagram of *DCA* approach [40]. The determination of each node involves three consecutive steps:



Begin
getMap(Map); getSource(Source); getTarget(Target);
Path[0]=Source; CPos=Source; MovePoint[]=generateMovePoint(Obs[]); seg=calculateSegment(CPos, Target); Obs[]=seg∩Obsj[], j={1..m}; Distance[]=calculateDistance(Obs[]); index=Arg(Min(Distance[])); Obs=Obs[index];
{P₁,P₂}∈{Seg₁, Seg₂}?
{P₁,P₂}∈{Seg₂, Seg₃}?
{P₁,P₂}∈{Seg₃, Seg₄}?
{P₁,P₂}∈{Seg₁, Seg₄}?
{P₁,P₂}∈{Seg₁, Seg₃}?
{P₁,P₂}∈{Seg₂, Seg₄}?
Case1
Case2
Case3
Case5
Case6
Case7
node=MovePoint₁;
node=MovePoint₂;
node=MovePoint₃;
node=MovePoint₄;
Path[i]=Target;
End
collide(CPos,node)?
Path[i]=node; CPos=Path[i]; i++;
generateIntermediateNodes(newNodes); add(newNodes, Path[]); i++;
Yes
No
Begin
getPath(Path); getNodeNumber(n);
i=0; j=i+2;
N₁=Path[i]; N₂=Path[j];
[N₁,N₂]∩Obsₖ={P₁,P₂}, k=1…m?
safetyDistance?
removeNodes(Nₕ, h<k<l);
j++;
j=n?
i++;
j=n?
End

(a) Overall diagram of the *DCA* (b) Diagram of nodes reduction

Fig. 2. Generation, optimization and reduction of nodes numbers of the path

a) Step one: The line segment $[CPos, Target]$ defined by $CPos$ and $Target$ is calculated. Then, the obstacles set $Obs\{\}$ crossed by this segment is determined by

$$Obs\{\} = [CPos, MovePoint] \cap Obs\_Seg_{tj}, \forall t \in \{1 \dots 4\}, \forall j \in \{1 \dots m\} \tag{11}$$

b) Step two: Once the set $Obs\{\}$ is determined, the closest obstacle to the robot $Obs_k, k \in \{0 \dots lenght(Obs\{\})\}$ is selected, and $MovePoint_i, i \in \{1 \dots 4\}$ of this obstacle are calculated. After that, intersection points $\{P_1, P_2\} = [CPos, Target] \cap Obs\_Seg_{tk}, t \in \{1 \dots 4\}$ are found. Depending on the locations of $\{P_1, P_2\}$ on the segments of Obs_k, the best $MovePoint$ is selected to avoid Obs_k. Table 2 shows the different cases and the corresponding actions.
c) Step three: In this last step, two cases exist:

- $[CPos, MovePoint] \cap Obs_j = \varnothing, \forall j \in \{1 \dots m\}$: The path and $CPos$ are updated with the determined $MovePoint_i (Path[i] = MovePoint_i, CPos = MovePoint_i)$; then, the process is repeated from the first step.

Table 2. Intersection cases and related actions

1. $\{P_1, P_2\} \in \{Seg_1, Seg_2\}$	Select $MovePoint_2$.
2. $\{P_1, P_2\} \in \{Seg_2, Seg_3\}$	Select $MovePoint_3$.
3. $\{P_1, P_2\} \in \{Seg_3, Seg_4\}$	Select $MovePoint_4$.
4. $\{P_1, P_2\} \in \{Seg_1, Seg_4\}$	Select $MovePoint_1$.
5. $\{P_1, P_2\} \in \{Seg_1, Seg_2\}$	Select $MovePoint_2$.
6. $\{P_1, P_2\} \in \{Seg_1, Seg_3\}$	Select the point according to $\{P_1, P_2\}$ (Figure 3a of [40])
7. $\{P_1, P_2\} \in \{Seg_2, Seg_4\}$	Select the point according to $\{P_1, P_2\}$ (Figure 3b of [40]).

- $[CPos, MovePoint] \cap Obs\_Seg_{tj} \neq \varnothing$: A collision occurs with obstacle Obs_j, the process restarts from the first step to generate nodes from $CPos$ until reaching $MovePoint$, while avoiding Obs_j ($MovePoint$ is considered as a new sub-destination). These steps are repeated until no obstacle exists between $CPos$ and $Target$.

Fig. 2a of [46] presents an illustrative example to clarify the *DCA* returning the shortest path in a complex W-space.

3.1.2. Reduction of nodes in the generated path

The algorithm given in Fig. 2b minimizes the number of nodes in the constructed paths. It relies on verifying the collision with obstacles for a specific segment $[N_h, N_l]$ composed of nodes $\{N_h, N_l\}$. If $([N_h, N_l] \cap Obs_j = \varnothing) \wedge (l \geq h + 2), \forall l, h \in \{1 \dots n\}, j \in \{1 \dots m\}$, the path may be further shortened by eliminating N_k where $\{h < k < l\}$ (Figures 2b and 2c of [46]).

3.2. Dynamic obstacles avoidance and traps escape

Because of the dynamic nature of the evolving W-spaces (e.g., industrial environments, houses, etc.), it is crucial to consider the pop-up obstacles as the robot moves through the nodes generated during the optimization phase.

This subsection presents a navigation strategy designed to avoid dynamic obstacles and escape traps. Our focus is particularly on FL due to its ability to make decisions and emulate human reasoning, despite the lack of precise information and accurate model equations in complex environments [41]. The objective is to design an FLC to safely guide the robot through multiple sub-goals (each node generated in the offline phase is considered as a sub-goal that the robot needs to reach) without colliding with dynamic obstacles or becoming trapped. Fig. 3 gives the diagram of the developed *EFLC* that ensures three behaviors (*i*) *Obstacle Avoidance*, (*ii*) $Target$ *Reaching*, and (*iii*) *Trap Escaping*. It has seven input variables:

- *Distance to* $Target$:

$$Dist - Target = \sqrt{(X_r - X_T)^2 + (Y_r - Y_T)^2} \tag{12}$$

- *Angle between the robot heading and the vector connecting the robot center to* $Target$:

$$Error - Angle = \arctan\left(\frac{Y_T - Y_r}{X_T - X_r}\right) - \arctan\left(\frac{Y_r}{X_r}\right) \tag{13}$$

- *Distance to closest obstacles in Front area (Obs_F), Right area (Obs_R) and Left area (Obs_L) of the robot*: sensors of right and left areas set a tilt angle of 30° to the left or right for better obstacles detection without the influence of other areas (Fig. 3).
- *Distance to closest obstacles on Left ($Left - Obs$) and Right ($Right - Obs$) side of the robot*: the approach distinguishes between left/right sides and left/right areas (Fig. 3).

The outputs of the *EFLC* approach are *Right* and *Left* velocities (V_R, V_L) calculated depending on the degree of the robot closeness to obstacles and $Target$.

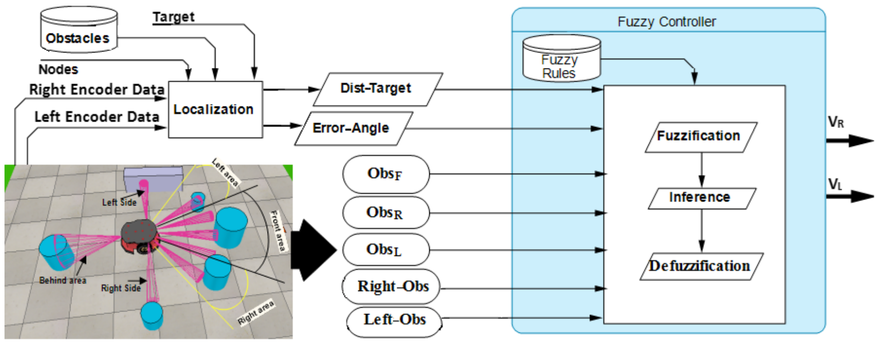


Fig. 3. Block diagram of the proposed *EFLC*

3.2.1. Membership functions and linguistic terms

Triangular and Gaussian functions are chosen to represent the FL sets of the *EFLC*. The tuning of the membership functions is realized according to the procedure detailed in [46]. Figs. 4a and 4b of [46] show the membership functions for the input and the output variables, respectively.

3.2.2. Human-reasoning-based navigation strategy

The proposed safe navigation strategy is based on the analysis of human reasoning in dynamic environments to avoid obstacles while pursuing multiple goals. Three navigation behaviors ($Target$ *Reaching*, *Obstacle Avoidance* and *Trap Escaping*) that mimic human reasoning have been implemented. Table I of [46] describes the human-reasoning-based transformed into efficient FL rules. For more details, please refer to [46].

4. Performance Evaluation of *DCA-EFLC* approach

A series of results assesses the efficiency of the developed *DCA-EFLC* approach: (*i*) results returned by *DCA* for a complex self-designed environment cluttered with many static obstacles; (*ii*) comparisons of *DCA* with other powerful algorithms such as GA, RRT, ACO, Dijkstra and QL in terms of *path length*, *runtime* and *safety*; and (*iii*) a validation is performed using *V-REP* software [47] to show the strength of *DCA-EFLC*. The *Mamdani* model is utilized as an FL inference engine; the *MIN-MAX* method is used throughout the inference process. The defuzzification phase employs the *Center of Gravity* method. Simulations are run on a PC with Intel Core $i3$ 3.30GHz having 8GB RAM and *64-bit Windows*.

4.1. Results of *DCA* on a self-designed complex map

A self-designed complex map is considered to evaluate the feasibility and efficacy of *DCA*; obstacles placement creates numerous narrow passages in between. Fig. 4 shows the obtained optimal and collisionless path with $Source(25, 775)$, $Target(775, 25)$, and $SafetyDist = 10units$. Additionally, $runtime = 0.062s$, which attests that the proposed approach is able to return a good solution in a very short time.

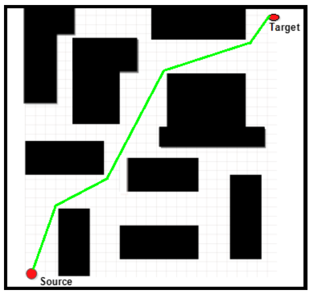


Fig. 4. Shortest path in self-designed complex map cluttered with many statics obstacles

4.2. Comparison of *DCA* with metaheuristic algorithms

Results obtained by the proposed *DCA* are compared with several well-known powerful planners in terms of *path length*, *safety* and *runtime*:

- **Comparison with *memEAPF***: It is made with *membrane Evolutionary Artificial Potential Field* (*memEAPF*) algorithm on some maps [42]. Authors propose many different maps; however, the validation is only made with the most complex maps with many narrow passages. This gives the robot several choices to get to $Target$. Table 3 gives the obtained results. Similarly, Fig. 5 shows the optimized paths generated by the proposed *DCA*.

Table 3. Comparison of *DCA* with *memEAPF* [42] on M_5, M_7, M_8 and M_{12} environments (bold numbers refer to best solutions, $IP(\%)$: *Improvement Percentage*)

Maps	**Path length (unit)**			**Runtime (s)**		
	memEAPF	**DCA**	**IP (%)**	**memEAPF**	**DCA**	**IP (%)**
Map_5	**6.37**	6.50	-2.04	3.16	**0.010**	99.68
Map_7	**7.72**	8.30	-7.51	3.06	**0.031**	98.98
Map_8	8.28	**7.10**	14.25	3.00	**0.031**	99.96
Map_{12}	9.24	**9.08**	1.730	3.19	**0.031**	99.02

- **Comparison with *GA* and *RRT***: Three variants of GA proposed in [43], [44] and [45] evoked in a complex grid map with 11 static obstacles are used. Since the computation time is not mentioned, comparison is only made in terms of *path length* (Figs. 6a-6d). Fig. 6e shows comparisons of *DCA* (red), *GA-PCHIP* (*Piecewise Cubic Hermite Interpolating Polynomial*) approach [9] (black), and *RRT-PCHIP* approach (green) [39].

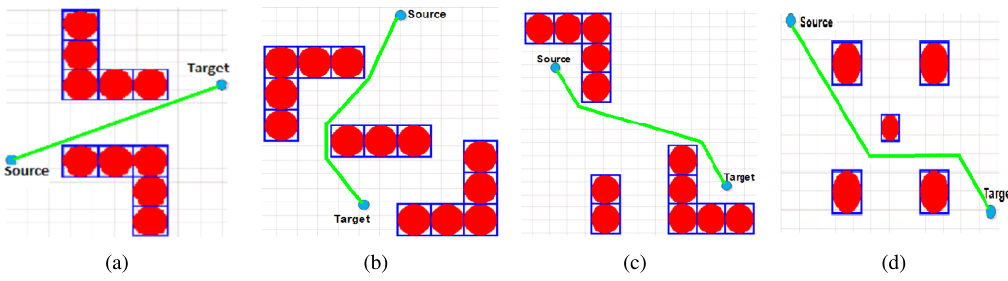


Fig. 5. Simulation results on the maps tested in [42]

- **Comparison with *ACO* and *Dijkstra***: Fig. 6f compares the paths obtained by the proposed *DCA* (black) with those of *Dijkstra* (blue), *improved GA* (green) and *ACO* (red) provided in [10].

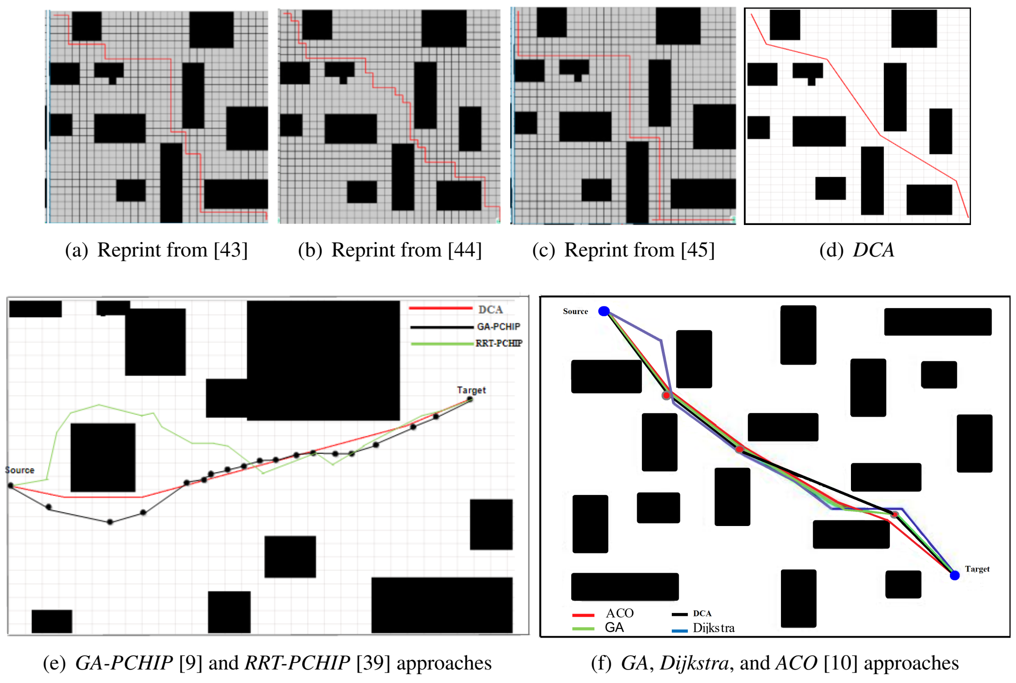


Fig. 6. Comparison of *DCA* with many variant of *GA*, *RRT*, *ACO*, and *Dijkstra*

4.3. Discussions of results obtained by *DCA*

DCA performs well in all the considered W-spaces no matter the obstacles arrangement. Another prominent trend observed in the metric *computation time* is that *DCA* makes a good tradeoff between minimizing the *path length* and *convergence speed*.

Table 3 compares results (*path length*, *runtime*) delivered by *DCA* with *memEAPF*. The nondeterministic nature of *memEAPF* imposes several independent runs on each map. It is easy to see that *DCA* significantly improves the *path length* for the two last tests with an IP of 14.25% and 1.73%, respectively. For both tests, *memEAPF* produced the best results. Fig. 5 illustrates the effectiveness of *DCA* in finding best quality paths in terms of *length* and *safety*. *DCA* remarkably improved the runtime by 98.26% compared to *memEAPF*. Therefore, it is deduced that *DCA* typically outperforms *memEAPF* since it requires less runtimes to yield better paths.

Simulation results shown in Figs. 6a-6d are obtained by running the *GA* approaches 40 times for each problem [45]. On the contrary, *DCA* is executed only once. In addition, in terms of *path length*, *DCA* generated the best path while minimizing zigzags present in the other paths.

More satisfactory results can be seen in Fig. 6e in terms of *path length* and *safety*. It is evident that *DCA* covers shorter distance to reach $Target$ compared to *GA-PCHIP* or *RRT-PCHIP*. As in the previous case, *GA-PCHIP* [9] and *RRT-PCHIP* [39] are executed 10 times each; whereas, *DCA* is executed only once. The results given in Fig. 6f show the best paths returned by all the approaches. In terms of *path length*, *Dijkstra* obviously gives the worst performance. Although this algorithm is efficient for global planning, *path length* and *smoothness* of the resulting paths are obviously poor. The red path generated by *ACO* is optimal; unfortunately, obstacles vertices are taken as the nodes, which may cause collisions (i.e., less safety). In terms of *safety* and *path length*, *DCA* and *GA* provided the best results.

Generally, the *safety* is obtained in the *GA* approaches by adding a penalty to the path length (when getting closer or across an obstacle) or by defining a reward in the fitness function. This procedure makes it hard to ensure the safety if the robot dimensions change, for example. In both cases, the sizes of robot or obstacles are not taken into account and the optimization speed is deteriorated. However, *DCA* simply checks this safety using $SafetyDist$ while estimating its value based on the robot dimensions.

The proposed *DCA* exhibits attractive features, such as *ability to find shortest solutions* (zero-failure rates), *high stability* (zigzags reduction), *low running cost* (*DCA* is significantly faster), and *collisionless paths* (high safety).

4.4. Validation and simulation of *DCA-EFLC* using *V-REP* software

The strength of the proposed *DCA-EFLC* approach is shown via a differentially-driven mobile robot (*Pioneer P3-DX*) evolving inside a dynamic environment (obstacles are added after the optimization phase) on the *V-REP* software. The generated path consists of two nodes at position $Node_1(-1.8, 1.2)$ and $Target(-0.15, 3.5)$. Further, $Source(-1.5, 0.07)$ is placed inside a *U-shape blocking region* (Fig. 7a). The robot must escape from this trap situation, arrive at $Node_1$ and then reach $Target$ while avoiding the dynamic obstacles.

Fig. 7 attests the strong performance and success of *EFLC* in guiding the robot as it moves along the route nodes. In fact, the robot can reach the route nodes generated by *DCA* ($Node_1 \ldots Target$) and avoid the dynamic obstacles. When it finds a *U-shape blocking zone*, it rotates on itself and chooses the direction toward $Target$ (to the left) to escape. Following that, the robot determines whether or not an obstacle is ahead. The *EFLC* approach can decide on the forthcoming step by evaluating the collision risk for the next position. As a result, the *EFLC* directs the robot to avoid the obstacle (Fig. 7c). The robot moves on to the subsequent sub-goal after completing the first ($Node_1$) as shown in Fig. 7d. The robot has two options, as shown in Fig. 7e, and the *EFLC* determines whether the right or left tight narrow passage is the best and easiest to traverse. The *EFLC* efficiently leads the robot to select the simpler passage, the near-optimal path, and safely traverse the narrow passage (Figs. 7f and 7g). Finally, the robot applies the same reasoning to the next node until it reaches $Target$ (Fig. 7h).

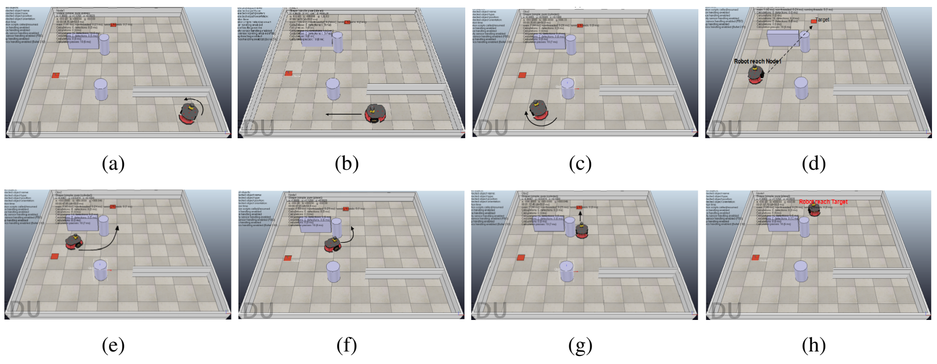


Fig. 7. Sequence showing the *V-REP simulator* implementing the proposed *EFLC*

Many conclusions can be drawn from the results of Figs. 4-7. The proposed *DCA-EFLC* approach combines both path optimization and motion control, which together ensure stability during navigation. The system consistently displays a good performance and exhibits superior performance in terms of *path length*, *runtime*, and *safety*. Indeed, *DCA* is constructive and only requires a very low computation time to generate better solutions. Unlike metaheuristic algorithms that strongly rely on the initial solution and their parameters setting, the *DCA* planner does not depend on any parameter. Further, *DCA* is deterministic, and has the ability to adapt to environment changes while exploiting the developed *EFLC* human reasoning-based approach to avoid dynamic obstacles. *DCA* guarantees the generation of feasible short paths that avoid static obstacles in the environment, contributing to the stability of the robot trajectory, *EFLC* provides a robust motion control mechanism that enables the robot to smoothly navigate through the environment, avoid dynamic obstacles and escape traps. The *EFLC* incorporates human-like reasoning, allowing for adaptability and resilience in complex and changing environments. This capability ensures that the robot maintains stability even when faced with dynamic obstacles.

5. Conclusions and Future Work

This paper contributed to the literature on mobile robotics by proposing a navigation and control system that integrates effective path optimization and motion control capabilities. The navigation process starts with the path planner, where a *Deterministic Constructive Algorithm* (*DCA*) builds the best path to get the robot from its current position to the goal position. Afterwards, while the robot is following this planned path, the *Efficient Fuzzy Logic Controller* (*EFLC*) continuously guides the robot around the dynamic obstacles while calculating the right and left velocities required to drive safely along the intended path. Simulation results on various navigation maps and comparisons with other planners demonstrated the effectiveness and superiority of the developed *DCA*. It was found that this approach returned good solutions and outperformed other approaches in terms of *path length*, *safety* and *runtime*. The proposed *EFLC*, which mimics human reasoning, is evaluated via an experimental validation on *V-REP software*. Results attested the strength of the control strategy in dynamic workspaces. Future work will investigate how to apply and adapt the vision system developed in [9] to model the robot surroundings through images processing techniques. Finally, since the efficiency of the *EFLC* approach strongly depends on its rules, it is necessary to tune them to find the most suitable behaviors.

Acknowledgement

The authors would like to thank the anonymous reviewers for their valuable recommendations and comments, as they led to considerably enhance the paper quality.

References

[1] L. ZUO, Q. GUO, X. XU and H. FU, *A hierarchical path planning approach based on A* and least-squares policy iteration for mobile robots*, Neurocomputing **170**, pp. 257–266, 2015.

[2] M. NAZARAHARI, E. KHANMIRZA and S. DOOSTIE, *Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm*, Expert Systems with Applications **115**, pp. 106–120, 2019.

[3] A. HENTOUT, A. MAOUDJ, D. YAHIAOUI and M. AOUACHE, *RRT-A*-BT approach for optimal collision-free path planning for mobile robots*, Algerian Journal of Signals and Systems **4**(2), pp.39–50, 2019.

[4] S. KARAMAN and E. FRAZZOLI, *Sampling-based algorithms for optimal motion planning*, International Journal of Robotics Research **30**(7), pp. 846–894, 2011.

[5] P. RAJA and S. PUGAZHENTHI, *Optimal path planning of mobile robots: A review*, International Journal of Physical Sciences **7**(9), pp. 1314-1320, 2012.

[6] K. NADERI, J. RAJAMÄKI, and P. HÄMÄLÄINEN, *RT-RRT*: A real-time path planning algorithm based on RRT*, Proceedings of 8^{th} ACM SIGGRAPH Conference on Motion in Games, Paris, France, pp. 113–118, 2015.

[7] A. TUNCER and M. YILDIRIM, *Dynamic path planning of mobile robots with improved genetic algorithm*, Computers & Electrical Engineering **38**(6), pp. 1564–1572, 2012.

[8] A. H. KARAMI and M. HASANZADEH, *An adaptive genetic algorithm for robot motion planning in 2D complex environments*, Computers & Electrical Engineering **43**, pp. 317–329, 2015.

[9] A. BAKDI, A. HENTOUT, H. BOUTAMI, A. MAOUDJ, O. HACHOUR and B. BOUZOUIA, *Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control*, Robotics and Autonomous Systems **89**, pp. 95–109, 2017.

[10] R. XI, Z. YANG, L. TANG and Z. WANG, *Robust GA based global path planning for IoP oriented mobile robot*, Proceedings of 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, Guangzhou, China, pp. 1962–1969, 2018.

[11] D. KARABOGA and B. BASTURK, *A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm*, Journal of Global Optimization **39**, pp. 459–471, 2007.

[12] M. A. CONTRERAS-CRUZ, V. AYALA-RAMIREZ and U. H. HERNANDEZ-BELMONTE, *Mobile robot path planning using Artificial Bee Colony and evolutionary programming*, Applied Soft Computing **30**, pp. 319–328, 2015.

[13] R. K. MANDAVA, S. BONDADA and P. R. VUNDAVILLI, *An optimized path planning for the mobile robot using potential field method and PSO algorithm*, in Soft Computing for Problem Solving, J. Bansal, K. Das, A. Nagar, K., Deep and A. Ojha, Eds., Springer, Singapore, Advances in Intelligent Systems and Computing **817**, pp. 139–150, 2019.

[14] G. LI and W. CHOU, *Path planning for mobile robot using self-adaptive learning particle swarm optimization*, Science China Information Sciences **61**(5), 2018, paper 052204.

[15] Y. ZHANG, D.-W. GONG and J.-H. ZHANG, *Robot path planning in uncertain environment using multi-objective particle swarm optimization*, Neurocomputing **103**, pp. 172–185, 2013.

[16] A. H. GANDOMI, X.-S. YANG and A. H. ALAVI, *Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems*, Engineering with Computers **29**, pp. 17–35, 2013.

[17] P. DAS, H. BEHERA, P. JENA and B. PANIGRAHI, *Multi-robot path planning in a dynamic environment using improved gravitational search algorithm*, Journal of Electrical Systems and Information Technology **3**(2), pp. 295–313, 2016.

[18] R.-E. PRECUP, R.-C. DAVID, E. M. PETRIU, S. PREITL and A. S. PAUL, *Gravitational search algorithm-based tuning of fuzzy control systems with a reduced parametric sensitivity*, in Soft Computing in Industrial Applications, A. Gaspar-Cunha, R. Takahashi, G. Schaefer and L. Costa, Eds., Springer, Berlin, Heidelberg, Advances in Intelligent and Soft Computing **96**, pp. 141–150, 2011.

[19] X. DAI, S. LONG, Z. ZHANG and D. GONG, *Mobile robot path planning based on ant colony algorithm with A* heuristic method*, Frontiers in Neurorobotics **13**, paper 15, 2019.

[20] J. ZHAO, D. CHENG and C. HAO, *An improved ant colony algorithm for solving the path planning problem of the omnidirectional mobile vehicle*, Mathematical Problems in Engineering **2016**, 2016, paper 7672839.

[21] C. MA, H. HUANG, Q. FAN, J. WEI, Y. DU and W. GAO, *Grey wolf optimizer based on aquila exploration method*, Expert Systems with Applications **205**, 2022, paper 117629.

[22] C.-A. BOJAN-DRAGOS, R.-E. PRECUP, S. PREITL, R.-C. ROMAN, E.-L. HEDREA and A.-I. SZEDLAK-STINEAN, *GWO-based optimal tuning of type-1 and type-2 fuzzy controllers for electromagnetic actuated clutch systems*, IFAC-PapersOnLine **54**(4), pp. 189–194, 2021.

[23] R.-E. PRECUP, R.-C. DAVID, R.-C. ROMAN, E. M. PETRIU and A.-I. SZEDLAK-STINEAN, *Slime mould algorithm-based tuning of cost-effective fuzzy controllers for servo systems*, International Journal of Computational Intelligence Systems **14**(1), pp. 1042–1052, 2021.

[24] R.-E. PRECUP, S. PREITL, E. M. PETRIU, J. K. TAR, M. L. TOMESCU and C. POZNA, *Generic two-degree-of-freedom linear and fuzzy controllers for integral processes*, Journal of the Franklin Institute **346**(10), pp. 980–1003, 2009.

[25] R.-E. PRECUP, S. PREITL, I. J. RUDAS, M. L. TOMESCU and J. K. TAR, *Design and experiments for a class of fuzzy controlled servo systems*, IEEE/ASME Transactions on Mechatronics **13**(1), pp. 22–35, 2008.

[26] J. LIU, J. YANG, H. LIU, X. TIAN and M. GAO, *An improved ant colony algorithm for robot path planning*, Soft Computing **21**(19), pp. 5829–5839, 2017.

[27] A. MAOUDJ and A. L. CHRISTENSEN, *Q-learning-based navigation for mobile robots in continuous and dynamic environments*, Proceedings of 2021 IEEE 17^{th} International Conference on Automation Science and Engineering, Lyon, France, pp. 1338–1345, 2021.

[28] G.-Z. TAN, H. HE and S. AARON, *Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm*, Journal of Central South University of Technology **13**(1), pp. 80–86, 2006.

[29] R. KALA, A. SHUKLA and R. TIWARI, *Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning*, Artificial Intelligence Review **33**, pp. 307–327, 2010.

[30] A. MAOUDJ and A. HENTOUT, *Optimal path planning approach based on Q-learning algorithm for mobile robots*, Applied Soft Computing **97**, paper 106796, 2020.

[31] E. S. LOW, P. ONG and K. C. CHEAH, *Solving the optimal path planning of a mobile robot using improved Q-learning*, Robotics and Autonomous Systems **115**, pp. 143–161, 2019.

[32] L. CHANG, L. SHAN, C. JIANG and Y. DAI, *Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment*, Autonomous Robots **45**(1), pp. 51–76, 2021.

[33] F. KAMIL, T. S. HONG, W. KHAKSAR, N. ZULKIFLI and S. A. AHMAD, *An ANFIS-based optimized fuzzy-multilayer decision approach for a mobile robotic system in ever-changing environment*, International Journal of Control, Automation and Systems **17**(1), pp. 253–266, 2019.

[34] J. YUAN, H. WANG, C. LIN, D. LIU and D. YU, *A novel GRU-RNN network model for dynamic path planning of mobile robot*, IEEE Access **7**, pp. 15140–15151, 2019.

[35] C.-F. JUANG and Y.-C. CHANG, *Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments*, IEEE Transactions on Fuzzy Systems **19**(2), pp. 379–392, 2011.

[36] R.-C. ROMAN, R.-E. PRECUP, E.-L. HEDREA, S. PREITL, I. A. ZAMFIRACHE, C.-A. BOJAN-DRAGOS and E. M. PETRIU, *Iterative feedback tuning algorithm for tower crane systems*, Procedia Computer Science **199**, pp. 157–165, 2022.

[37] M. ALGABRI, H. MATHKOUR, H. RAMDANE and M. ALSULAIMAN, *Comparative study of soft computing techniques for mobile robot navigation in an unknown environment*, Computers in Human Behavior **50**, pp. 42–56, 2015.

[38] A. MAOUDJ, A. HENTOUT, B. BOUZOUIA and R. TOUMI, *On-line fault-tolerant fuzzy-based path planning and obstacles avoidance approach for manipulator robots*, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **26**(5), pp. 809–838, 2018.

[39] A. HENTOUT, A. MAOUDJ, D. GUIR, S. SAIGHI, M. A. HARKAT, M. Z. HAMMOUCHE and A. BAKDI, *Collision-free path planning for indoor mobile robots based on rapidly-exploring random trees and piecewise cubic hermite interpolating polynomial*, International Journal of Imaging and Robotics **19**(3), pp. 74–97, 2019.

[40] A. MAOUDJ, A. HENTOUT, A. L. CHRISTENSEN and A. KOUIDER, *A fast constructive path planning algorithm for mobile robot navigation*, Proceedings of 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vasteras, Sweden, pp. 1–8, 2021.

[41] A. HENTOUT, A. MAOUDJ and M. AOUACHE, *A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots*, Artificial Intelligence Review **56**(3), pp. 3369–3444, 2023.

[42] U. OROZCO-ROSAS, O. MONTIEL and R. SEPULVEDA, *Mobile robot path planning using membrane evolutionary artificial potential field*, Applied Soft Computing **77**, pp. 236–251, 2019.

[43] M. ALAJLAN, A. KOUBAA, I. CHAARI, H. BENNACEUR and A. AMMAR, *Global path planning for mobile robots in large-scale grid environments using genetic algorithms*, Proceedings of 2013 International Conference on Individual and Collective Behaviors in Robotics, Sousse, Tunisia, pp. 1–8, 2013.

[44] Z. QIONGBING and D. LIXIN, *A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimization problems*, Expert Systems with Applications **60**, pp. 183–189, 2016.

[45] C. LAMINI, S. BENHLIMA and A. ELBEKRI, *Genetic algorithm based approach for autonomous mobile robot path planning*, Procedia Computer Science **127**, pp. 180–189, 2018.

[46] A. HENTOUT, A. MAOUDJ and A. KOUIDER, *Efficient fuzzy-logic control of indoor mobile robots evolving in static and dynamic workspaces*, preprint, https://doi.org/10.21203/rs.3.rs-3182222/v1, 2023.

[47] COPPELIA ROBOTICS, https://www.coppeliarobotics.com/, 2023.