Multi-robot Cooperation and Path Planning Using Modified Cuckoo Search

Bandita Sahu, Pradipta Kumar Das, and Manas Ranjan Kabat

Abstract The paper proposes an innovative approach to solve the cooperation and path planning problem of multiple mobile robots in clutter environment. The main emphasis of the work lies in designing a multi-objective fitness function for stick-carrying robot pairs to compute a collision-free optimal path. The present context of the paper address the multi-robot cooperation and path planning of two pairs of stick-carrying robots that move from a predefined initial location to a pre-assumed goal position by carrying a stick at either end. The basic cuckoo search (CS) algorithm is modified concerning the step size and the scaling parameter at each step of movement of the robot pairs. The modified cuckoo search (MCS) algorithm is implemented with the robot pair to mimic the egg laying behavior of the cuckoo for producing the next generation solution. The proposed algorithm is validated using computer simulation and has been compared with other existing approaches such as ICFA, CS, SDA, and ABCO. Due to its simplicity and efficacy in terms of path optimality, the proposed algorithm produces an optimal solution both in the static and dynamic environment irrespective of the number of obstacles.

Keywords Stick-carrying robot · Multi-robot cooperation · Path planning · Path deviation · Performance

1 Introduction

Cooperation and path planning of multiple mobile robots have become a crucial research problem in the field of robotics [1]. Both the concepts have been paid much attention to solve several social and environmental issues such as object transportation

B. Sahu (✉)
Department of CSE, GIET University, Gunupur, Odisha, India
e-mail: bandita.sahu@giet.edu

P. Kumar Das
Department of IT, VSSUT, Burla, Odisha, India

M. Ranjan Kabat
Department of CSE, VSSUT, Burla, Odisha, India

R. Kumar et al. (eds.), *Next Generation of Internet of Things*, Lecture Notes in Networks and Systems 445, https://doi.org/10.1007/978-981-19-1412-6_31

in industry, robotics surgery and equipment transfer in healthcare system, detection of planetary movements in space, and disaster management in cluttered unreachable environment [2]. A robot may not be able to do these tasks solely and hence need the help of other robots present in the environment. When more than one robot work together to accomplish a task is called cooperation [3]. For completion of any task, the robots need to move from their position to the desired position. The point at which the transition starts is called as initial position, and the end point of transition is called goal position. Computing the path from the initial to the goal position is called path planning. A path for an autonomous robot can be computed easily. However, planning the path for cooperating robots is complex. Both the cooperation and the planning depend on the environment where the robot resides and has to complete the task. If the environment has motionless obstacle, it is referred to as static environment [4]. But, the environment having moving obstacle is referred to as dynamic environment. Static environment needs less number of computations as compared to the dynamic environment. The cooperation and path planning can be completed using two approaches such as local path planning of multiple mobile robots and global path planning [5]. In local path planning, the robots compute the path in a step-wise manner. It uses the concept of spatial object by considering a local area at each step. However, in global path planning, the robot computes it path as a whole in a known environment. The global approach may not guarantee to produce an optimal solution always [6].

A plethora of algorithms has been proposed to address the issue of path planning and cooperation of the mobile robots. The classical approaches such as potential field method [7], cell decomposition [8], and probabilistic road map [9] method solve the path planning problem of the mobile robots in static environment. But, these are unable to provide an optimal solution in dynamic environment, due to the inefficiency in dealing with the uncertainty [10]. Several measures have been taken for enhancing these classical approaches. Shortest distance algorithm [11] has been proposed to address the path planning problem in static environment without the presence of any obstacle. The authors have considered only an obstacle-free environment to compute the optimal path from the predefined start to the goal state. Hybridization of the existing approaches has also been proposed by using the combined benefits of individual algorithm such as BADE [12], PSO-DV [13], and IPSO-IGSA [14]. All these approaches address the path planning problem only, and the cooperation among the robot is overlooked. Meta-heuristic approaches such as particle swarm optimization (PSO) [15], artificial bee colony optimization (ABCO) [16], and imperialistic competitive firefly algorithm (ICFA) [17] have also been proposed to address the multi-robot path planning and cooperation problem. But they may not produce the optimal solution concerning time and path always as the solution quality depends upon the uncertainty of the environment most of the time. Several nature-inspired techniques such as intelligent water drop algorithm (IWD) [18], cuckoo search algorithm [19], social spider optimization [20], and bat algorithms [21] have been implemented by mimicking the behavior of some natural phenomena to address the path planning problem. However, the objective function of these algorithms is based on the path optimality only and the cooperation part is ignored.

The paper addresses the path planning and cooperation problem of multiple mobile robots in both static and dynamic environment. An optimal and collision-free path is computed using modified CS algorithm. The modification has been realized by tuning the step size and scaling factor dynamically. Two pairs of stick-carrying robot transit from the predefined starting position to a goal position. The main emphasis of the paper is furnished as (i) problem formulation for both path planning and cooperation, (ii) computation of a collision-free path for the robot pairs, (iii) validation of the algorithm by simulating it using programming in C language, and (iv) comparison of the proposed algorithm with other existing approaches.

The rest of the paper is furnished into four sections. Section 2 describes the basic CS algorithm along with the modification done to get the modified CS algorithm. In Sect. 3, the implementation of the proposed algorithm is done with the addressed problem. Section 4 explains the validation of the proposed algorithm through simulation. The conclusion and the future scope of the approach are included in Sect. 5.

2 Cuckoo Search Algorithm

A cuckoo search (CS) algorithm is a nature-inspired evolutionary algorithm inspired by the egg laying behavior of the cuckoo bird. The cuckoo lays their egg in the nest of other host birds. The basic CS algorithm is based on five principles such as (i) each cuckoo lays an egg in a randomly chosen nest, (ii) the egg laid in a host nest represents a solution, (iii) fixed numbers of host nests are assumed, (iv) the host nest having a quality egg is considered as the next generation, and (v) the probability if identifying or destroying the cuckoo's egg or the host bird abandon the best is either 0 or 1. The host nests are identified in a specified area of a fixed radius referred to as egg laying radius (ELR) and expressed as follows:

$$\mathrm{ELR} = E \times \frac{\mathrm{Num_{laid}}}{\mathrm{Num_{Total}}}\left(\mathrm{Max_{egg}} - \mathrm{Min_{egg}}\right) \tag{1}$$

where E represents the maximum radius, $\mathrm{Num_{laid}}$ and $\mathrm{Num_{Total}}$ denote the number of egg laid by a cuckoo and total number of eggs, respectively, and $\mathrm{Max_{egg}}$ and $\mathrm{Min_{egg}}$ holds the value of maximum and minimum number eggs than can be laid by a cuckoo. The following equation shows the basic step of computing the solution as a global walk of a current cuckoo $\mathrm{C_i}$ at time $\mathrm{t}+1$.

$$\mathrm{C_i(t+1)} = \mathrm{C_i(t)} + \alpha \oplus \mathrm{levy}(\beta) \tag{2}$$

where the probability distribution of levy flight is

$$\mathrm{levy} \sim u = \mathrm{t}^{\beta} \text{with } 1 < \beta < 3 \tag{3}$$

where α denotes the step size and β is the distribution parameter. The local walk for a cuckoo can also be expressed in the following equation

$$C_i(t+1) = C_i(t) + \alpha \oplus \left(C_i^{best}(t) - C_i^{best}(t)\right)\text{levy}(\beta) \tag{4}$$

where $C_i^{best}(t)$ denotes the best location in the history of the current cuckoo. The equation can further be written using the switching parameter of the host bird from its original nest. The equation for computing the next generation solution of the current cuckoo is represented using a heavy-side function $H(p_c - \varepsilon)$, the switching parameter p_s, and ε as a random value. The local walk is represented as follows.

$$C_i(t+1) = C_i(t) + \alpha \oplus H\left(p_s - \varepsilon\right) \oplus \left(C_i(t) - C_i^{best}(t)\right) \tag{5}$$

Each time a solution is generated, the fitness value is computed by considering the distance of the host nest from the target position and position of the obstacle present in the environment. It is computed as follows.

$$F(C_i) = \text{comp}(C_i, P_{obs}) \,\&\&\, D_{ELR} \,\&\&\, D_{i-g} \tag{6}$$

Such that

$$\text{comp}(C_i, P_{obs}) = \begin{cases} 1 & \text{if } C_i == P_{obs} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$D_{ELR} = \begin{cases} 1 & \text{if } (C_i - ELR) \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$D_{i-g} = \begin{cases} 1 & \text{if } \left(\left|C_i(t) - C_i^{best}(t)\right| + \left|C_i^{best}(t) - \text{goal}\right|\right) \\ & < \left(\left|C_i(t-1) - C_i^{best}(t-1)\right| + \left|C_i^{best}(t-1) - \text{goal}\right|\right) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The basic CS algorithm suffers from several pitfalls such as (i) low convergence rate, (ii) generation of same solution due to fixed parameters, (iii) repetitive calculation of host nest, and (iv) easy fall into the local optima. To resolve these issues, several measures have been considered with the basic approaches are varying step size, changing the scaling parameter and the penalty function dynamically. The following modifications have been considered in this article for the improvement of basic CS algorithm.

$$p_c = p_c + \frac{\text{rand} \times p_c}{n} \tag{10}$$

$$\alpha = \alpha \times \exp \frac{1}{n} \tag{11}$$

The scaling parameter for computing the probability has been computed using a random function that generates a value between 0 and 1 at each step of calculation. Further, the step size has also been modified to generate new step value in order to explore new host. The modified algorithm can be written as follows.

Algorithm 1: MCS (C)	
Input: Initial habitat C	
Output: best next generation solution C_i^{best}	
1	Initialize n, n, p_s, α, initial habitat C_i, $Max_{ITER} = 50$ and $t = 1$
2	Assume $C_i^{best} = C_i$
3	Evaluate the fitness of each nest $F(C_i)$ using Eq. 6
4	While ($t <= Max_{ITER}$)
5	Select a new nest C_j using Eq. 5
6	If $\lvert C_j - C_i \rvert > ELR$
7	Drop the selection and made a fresh one
8	else
8	Evaluate F (C_j)
9	If ($F_{profit}(C_i) \leq F_{profit}(C_j)$)
10	Set $C_i^{best} = C_j$
11	End if
12	Update p_c
13	Update α
14	increase iteration $t = t + 1$
15	End while

3 Implementation of Proposed Algorithm

The problem addressing the stick-carrying operation is formulated with a pair of robot (R_i,R_j) with the position vector (x_i,y_i) and (x_j,y), respectively, for both the robots. Each time the robot pairs compute a next best position and move with the same velocity and direction. For the cooperation, the velocity, direction of movement, and rotation if any are assumed to be same. At each step, the robot pairs update their position that is assumed as the solution as implemented with the MCS algorithm. The following algorithm describes the computation of the best path from the initial state to the goal state for both the robots.

Algorithm 2: Twin_MCS (C)

Input: Initial habitat C

Output: best next generation solution $C_{i,j}^{best}$

Initialize n, p_s, α, initial habitat C_i, $Max_{ITER} = 50$ and $t = 1$
Assume $C_{i,j}^{best} = C_{i,j}$
Evaluate the fitness of each nest $F(C_i)$ and $F(C_j)$using Eq. 6
While ($t <= Max_{ITER}$)
    Select a new nest C_i' and C_j' using Eq. 5
    If ($|C_i - C_i'| \geq$ ELR &&$|C_j - C_j'| >$ ELR)
        If ($|C_i' - C_j'| \neq l_{stick}$
            Drop the selection and made a fresh one
        else
            Evaluate F (C_i') andF (C_j')
            If ($F_{profit}(C_i) \leq F_{profit}(C_i')$ &&)$F_{profit}(C_j) \leq F_{profit}(C_j')$
                Set $C_{i,j}^{best} = C_{i,j}$
            End if
        End if
    Else
        Go to step 5 and do a fresh computation
    End if
    Update p_c
    Update α
    increase iteration $t = t + 1$
End while

It is assumed in the algorithm 2 that the robots are separated by a distance same as the stick length l_{stick}. The initial position for the robot pairs are selected from the assumed initial habitat C. The positions are selected as C_i and C_j. Using the equation as mentioned in the previous section, a new next solution is generated for both the robot. If the distance of the new positions is more than the ELR, it is rejected and a new solution is generated. Similarly, if the next position or the solutions generated by the robot pairs do not satisfy the constraint of stick length, the solution is dropped and a fresh calculation is made. The objective function formulated in this algorithm is based on two constraints such as distance constraint between the robot pairs and collision avoidance with the obstacle which is computed in terms of the fitness. It is assumed as a minimization problem that minimizes the path length between the pre-assumed start and goal state.

$$f = w_1 \times f_1 + w_2 \times f_2 \tag{12}$$

Where w_1, w_2 represents the weights and assigned with the value 0.5 and 0.5, f_1 indicates the objective function to compute the distance between the robot pairs and computed as follows.

$$f_1 = \begin{cases} 1 & \text{if} \quad (\left|C_i' - C_j'\right| \neq l_{stick} \\ \left|C_i' - C_j'\right| & \text{otherwise} \end{cases} \tag{13}$$

Similarly, the second objective function describes the fitness of the solution by comparing it with the obstacle position and represented as follows.

$$f_2 = \begin{cases} 1 & \text{if} \quad C_i' \neq C_{obs} \; and \; C_j' \neq C_{obs} \\ D - D_{th} & \text{otherwise} \end{cases} \tag{14}$$

where D_{th} denotes the threshold distance between the obstacle and the robot position, and D denotes the actual distance obtained between the obstacle and the computed position of the robot.

4 Simulation and Result

A simulation environment has been established as a 400 × 550 pixel area on a computer screen. The proposed algorithm along with its competitor has been executed using programming in C language. The environment has been set with 10 obstacles and two pairs of robots. The robots are denoted as circle of radius 5 pixel units. The initial and goal positions are represented with different color circle. The stick is represented as a rectangle of size 10 × 20 pixel area. Experiments have been performed with varying the number of obstacle as 8, 10, 12, and 15. The amount of path traveled by each algorithm like MCS, IVFA, CS, SDA, and ABCO has been computed and shown in Fig. 1. Further, the obstacles are allowed to move for realizing a dynamic environment. To generate a collision-free path concerning the obstacles in motion, all the above-mentioned algorithms are executed and the path traveled in unit pixels is recorded in Fig. 2.

Similarly, the number of turns required reaching the target in the static and dynamic environment is noted in Table 1. The number varies with respect to the number of obstacles. All the competing approaches are executed in the static environment incurred lesser number of turns as compared to the dynamic environment due to the motion of the obstacles. However, irrespective of the number of obstacles and environment, the proposed algorithm produces the optimal path with least number of turns as shown in Figs. 3 and 4. With less number of turns, the proposed algorithm requires less number of steps to reach its target. Figure 5 witnesses the supremacy of the proposed algorithm in terms of incurred steps in static environment. Similarly, the algorithms ICFA, SDA, CS, and ABCO have been executed along with the proposed algorithm in the dynamic environment and require lesser number of steps as shown in Fig. 6. Table 2 illustrates the superiority of the proposed algorithm as compared to other existing approaches in terms of steps. The amount

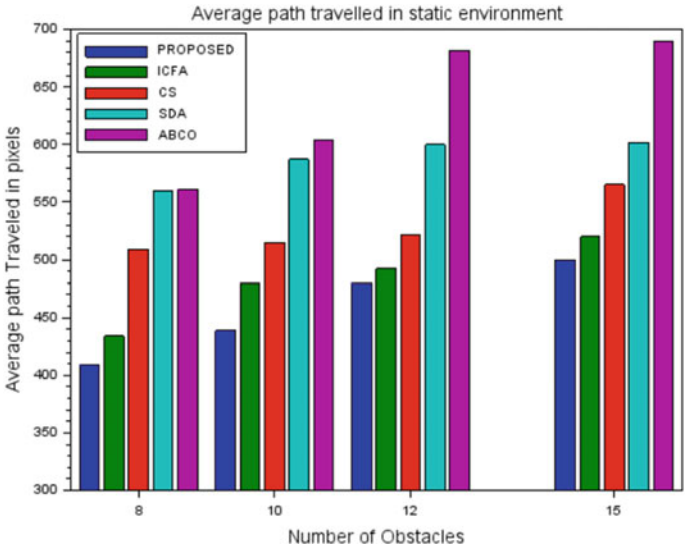

Fig. 1 Average path traveled in static environment

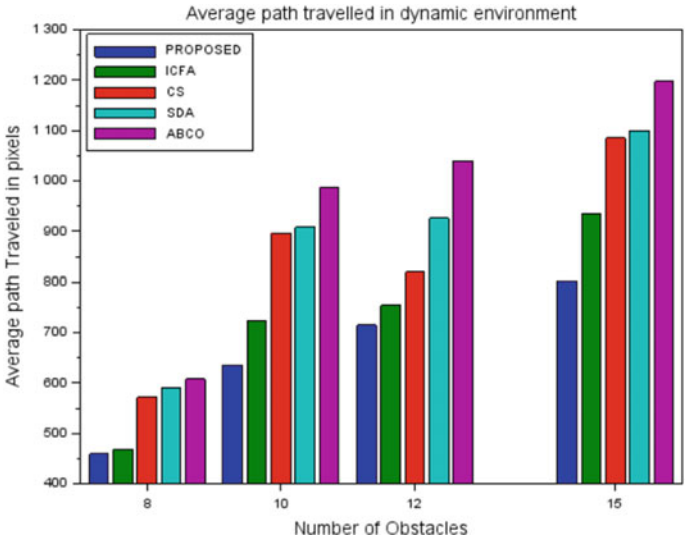

Fig. 2 Average path traveled in dynamic environment

of path deviated by each of the approaches is plotted in Figs. 7 and 8. In both the static and dynamic environment, the amount is minimal for the proposed algorithm.

Table 1 Number of turns require for the robot pairs to reach the target

Algorithms	Environment			
	Static		Dynamic	
	Robot pair1	Robot pair2	Robot pair1	Robot pair2
Proposed	7	11	9	12
ICFA	9	13	13	17
CS	12	16	15	20
SDA	14	19	19	23
ABCO	16	22	20	25

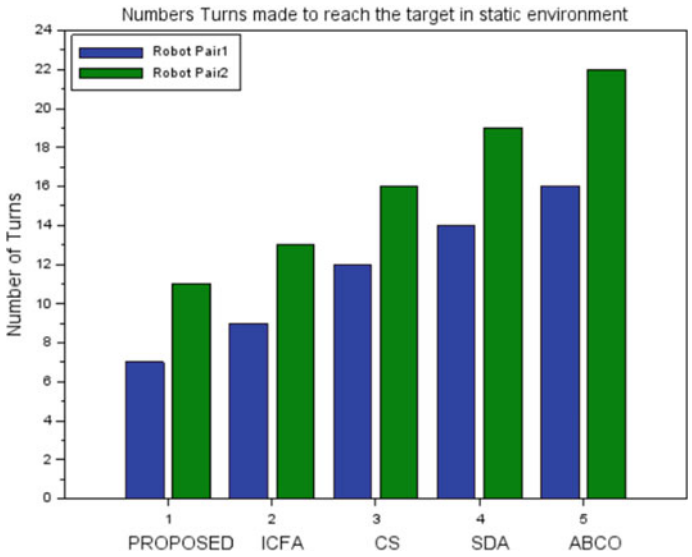

Fig. 3 Turns made by robot pairs in static environment

The overall performance for each of the competing algorithm has been computed by considering the number of steps, turns, and path traveled using the following equation.

$$\text{per} = \text{a*steps} + \text{b*turn} + \text{c*path} \tag{16}$$

In the above equation, the variables a, b, and c are assumed as the weighing parameters with the values 3, 2, and 1, respectively. The overall performance computed for

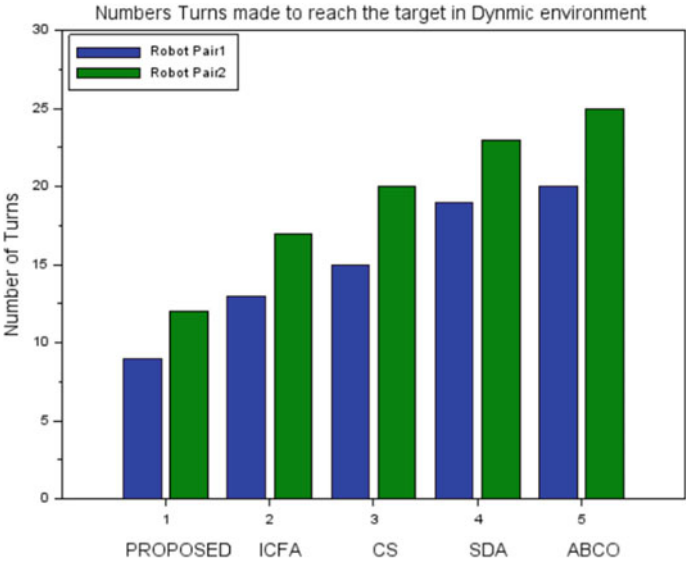

Fig. 4 Turns made by robot pairs in dynamic environment

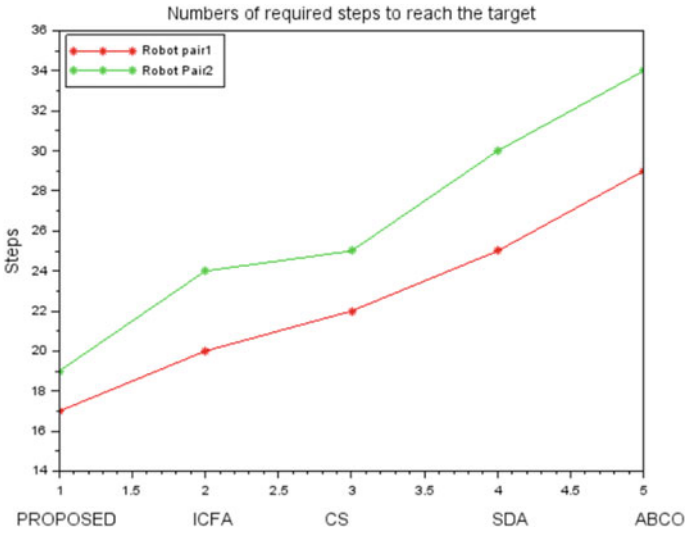

Fig. 5 Steps made by robot pairs in static environment

each of the algorithm is illustrated in Figs. 9 and 10 for the static and dynamic environment, respectively. In both the environment, the proposed algorithm outperforms the competing algorithm.

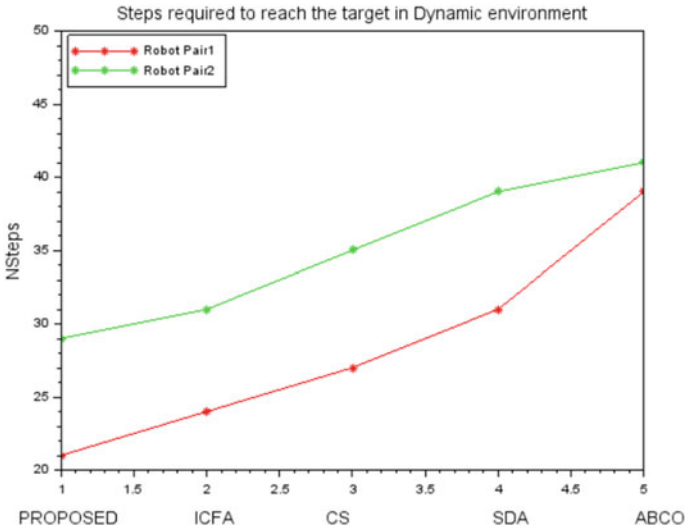

Fig. 6 Steps made by robot pairs in dynamic environment

Table 2 Number of steps require for the robot pairs to reach the target

Algorithms	Environment			
	Static		Dynamic	
	Robot pair1	Robot pair2	Robot pair1	Robot pair2
Proposed	17	30	21	29
ICFA	20	24	24	31
CS	22	25	27	35
SDA	25	30	31	39
ABCO	29	34	39	41

5 Conclusion

The primary focus of the paper is to captivate the problem formulation of multiple mobile robots to transport a stick from one position to another position. The research work is in the framework of nature-inspired techniques to mimic the egg laying behavior of cuckoo. The pitfall of the basic CS algorithm is resolved by tuning the step size and scaling parameter dynamically. The proposed algorithm is validated using computer simulation through programming in C language. The paper justifies the superiority of the proposed algorithm in terms of path optimality, collision avoidance, and performance in both the static and dynamic environment. Further research may be extended for more number of mobile robots to realize the cooperation among them as the proposed paper is limited to the synchrony of two robots only.

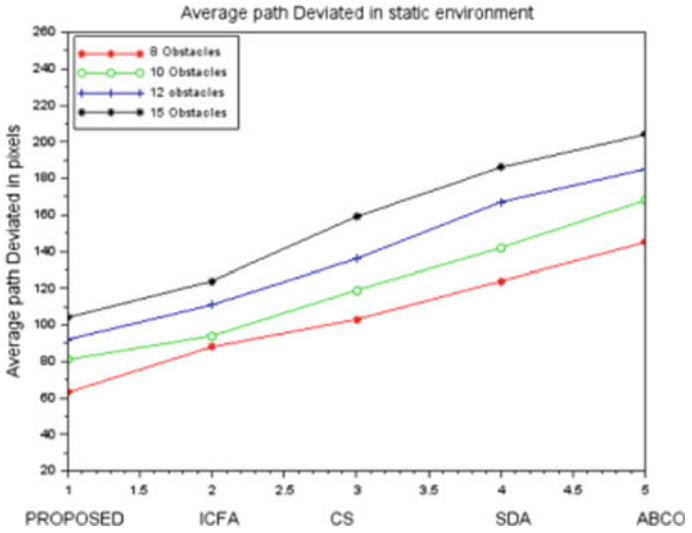

Fig. 7 Path deviated in static environment

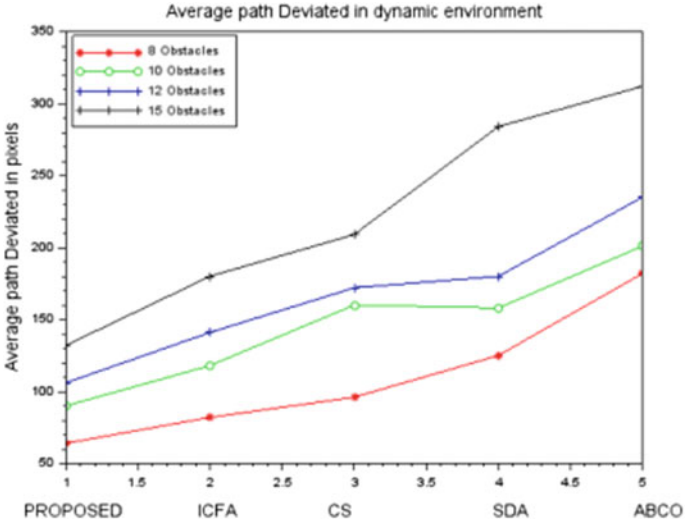

Fig. 8 Path deviated in dynamic environment

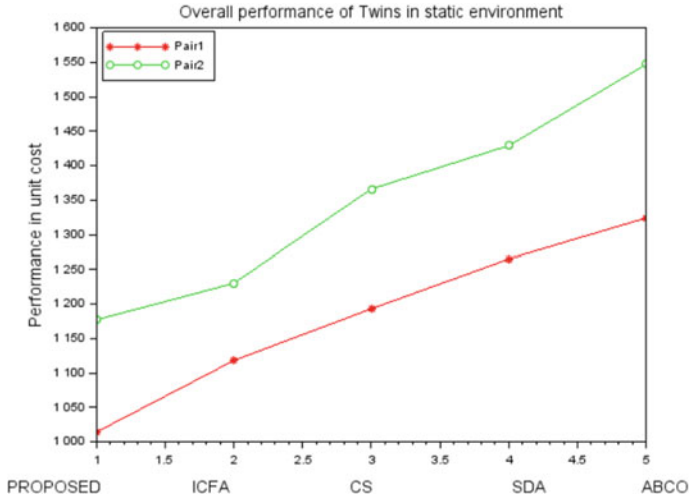

Fig. 9 Overall performance in static environment

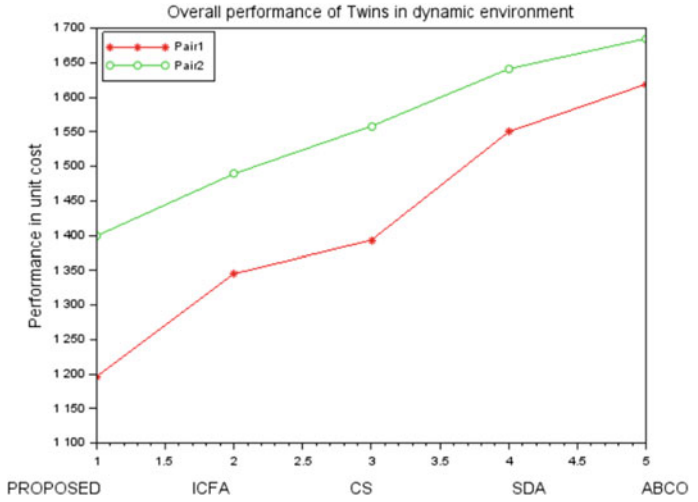

Fig. 10 Overall performance in dynamic environment

References

1. Olcay E, Schuhmann F, Lohmann B (2020) Collective navigation of a multi-robot system in an unknown environment. Robot Auton Syst 132:103604
2. Rajendran P et al (2021) Strategies for speeding up manipulator path planning to find high quality paths in cluttered environments. J Comput Inf Sci Eng 21(1):011009
3. Hamed O, Hamlich M (2020) Improvised multi-robot cooperation strategy for hunting a dynamic target. EAI Endorsed Trans Internet Things 6(24):e5

4. Chandrashekhar A, Himam Saheb S, Pavan Kishore ML (2021) Investigation of the static and dynamic path planning of mobile and aerial robots. Comput Networks Inventive Commun Technol, 1033–1044
5. Ab Wahab MN, Nefti-Meziani S, Atyabi A (2020) A comparative review on mobile robot path planning: classical or meta-heuristic methods? Annu Rev Control
6. Zhang H-y, Lin W-M, Chen A-X (2018) Path planning for the mobile robot: a review. Symmetry 10(10):450
7. Vadakkepat P, Tan KC, Ming-Liang W (2000) Evolutionary artificial potential fields and their application in real time robot path planning. In: 2000 Proceedings of the congress on evolutionary computation. CEC00 (Cat. No. 00TH8512). vol 1, IEEE
8. .Kloetzer M, Mahulea C, Gonzalez R (2015) Optimizing cell decomposition path planning for mobile robots using different metrics. In: 2015 19th international conference on system theory, control and computing (ICSTCC), IEEE
9. Santiago RM et al (2017) Path planning for mobile robots using genetic algorithm and probabilistic roadmap. In: 2017 IEEE 9th international conference on humanoid, nanotechnology, information technology, communication and control, environment and management (HNICEM). IEEE, 2017.
10. Liu Z et al (2020) Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties. IEEE Trans Autom Sci Eng
11. Ali AA et al (2016) An algorithm for multi-robot collision-free navigation based on shortest distance. Rob Auton Syst 75:119–128
12. Meng X, Gao X, Liu Y (2015) A novel hybrid bat algorithm with differential evolution strategy for constrained optimization. Int J Hybrid Inf Technol 8(1):383–396
13. Das PK et al (2016) A hybrid improved PSO-DV algorithm for multi-robot path planning in a clutter environment. Neurocomputing 207:735–753
14. Das PK, Behera HS, Panigrahi BK (2016) A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm Evol Comput 28:14–28
15. Song B, Wang Z, Zou L (2021) An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. Appl Soft Comput 100:106960
16. Das P et al (2015) Arduino based multi-robot stick carrying by artificial bee colony optimization algorithm. In: 2015 proceedings of the third international conference on computer, communication, control and information technology (C3IT), IEEE
17. Sadhu AK, Rakshit P, Konar A (2016) A modified imperialist competitive algorithm for multi-robot stick-carrying application. Robot Auton Syst 76:15–35
18. Salmanpour S, Motameni H (2014) Optimal path planning for mobile robot using intelligent water drops algorithm. J Intell Fuzzy Syst 27(3):1519–1531
19. Saraswathi M, Murali GB, Deepak BB (2018) Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm. Procedia Comput Sci 133:510–517
20. Kavitha S, Venkumar P (2020) A vibrant crossbreed social spider optimization with genetic algorithm tactic for flexible job shop scheduling problem. Meas Control 53(1–2):93–103
21. Guo J, Gao Y, Cui G (2015) The path planning for mobile robot based on bat algorithm. Int J Autom Control 9(1):50–60