

# Optimized cuckoo search algorithm using tournament selection function for robot path planning

*International Journal of Advanced  
Robotic Systems*

May-June 2021: 1–11

© The Author(s) 2021

Article reuse guidelines:

[sagepub.com/journals-permissions](https://sagepub.com/journals-permissions)

DOI: 10.1177/1729881421996136

[journals.sagepub.com/home/arx](https://journals.sagepub.com/home/arx)**Kaushlendra Sharma , Shikha Singh and Rajesh Doriya**

## Abstract

Acceptability of mobile robots in various applications has led to an increase in mobile robots' research areas. Path planning is one of the core areas which needs to be improvised at a higher level. Optimization is playing a more prominent role these days. The nature-inspired algorithm is contributing to a greater extent in achieving optimization. This article presents the modified cuckoo search algorithm using tournament selection function for robot path planning. Path length and Path time are the algorithm's parameters to validate the effectiveness and acceptability of the output. The cuckoo search algorithm's fundamental working principle is taken as the baseline, and the tournament selection function is adapted to calculate the optimum path for robots while navigating from its initial position to final position. The tournament selection function is replacing the concept of random selection done by the cuckoo search algorithm. The use of tournament selection overcomes local minima for robots while traversing in the configuration space and increases the probability of giving more optimum results. The conventional cuckoo search algorithm whose random selection mechanism may lead to premature convergence may fall into the local minima. The use of tournament selection function increases the probability of giving better results as it allows all the possible solution to take part in the tournament. The results are analysed and compared with other relevant work like cuckoo search algorithm and particle swarm optimization technique and presented in the article. The proposed method produced a better output in terms of path length and path time.

## Keywords

Cuckoo search, path length, path planning, tournament function

Date received: 21 November 2020; accepted: 20 January 2021

Topic Area: Mobile Robots and Multi-Robot Systems

Topic Editor: Nak-Young Chong

Associate Editor: Jonghoek Kim

## Introduction and related work

Path planning has been the issue in robotics system since the 1970s<sup>1,2</sup>; much contribution has been made in this particular field to make it more suitable and adaptable for the existing environment. Path planning problem for robots can be defined as the route adopted by the robots for reaching the destination from the source. The robot's total distance is known as the path length, and the problem starts taking shape from here on, it is required that how this path length could be minimized so that the robots should reach its destination covering a minimal distance. Again much

contribution is made to reduce the path length. To formulate the problem, the second parameter, which plays an important role is the path time, that is, the technique's time to derive

---

Department of Information Technology, National Institute of Technology Raipur, India

### Corresponding author:

Kaushlendra Sharma, Department of Information Technology, National Institute of Technology Raipur, Great Eastern Road, Amanaka, Raipur 492010, India.

Email: [ksharma.phd2017.it@nitrr.ac.in](mailto:ksharma.phd2017.it@nitrr.ac.in)



Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (<https://creativecommons.org/licenses/by/4.0/>) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (<https://us.sagepub.com/en-us/nam/open-access-at-sage>).

**Table 1.** List of applications where cuckoo search algorithms is applied with their respective parameters whose performance is improved using this algorithm.

Author	Year	Application	Parameters	RI	Reference
Valian et al.	2011	Neural network training for two benchmark classification problems	Accuracy, convergence rate	236	12
Layeb	2011	Knapsack problems	Reduced population size	166	13
Subotic et al.	2012	Unconstrained optimization problems	Parallelization, quality of service	20	14
Yang et al.	2012	Business optimization applications	Bankruptcy prediction	48	15
Yang and Deb	2013	Multiobjective design optimization	Disc brake design	587	16
Yildiz	2013	Manufacturing optimization problems	Maximizing profit rate	348	17
Basu and Chowdhury	2013	Micro grid power dispatch problems	Cost convergence	163	18
Ouaarab et al.	2014	Travelling salesmen problem	Minimizing overall cost	299	19
Mohanty and Parhi	2013	Path planning for mobile robot	Path length, path time	41	20
Bhandari et al.	2014	Satellite image segmentation	Entropy	284	21
Marichelvam et al.	2014	Hybrid flowshop scheduling problems	Minimize makespan	164	22
Goyal and Patterh	2014	Localization problem in wireless sensor network	Localization accuracy	45	23
Nguyen et al.	2014	Hydrothermal scheduling problems	Fuel cost function	81	24
Mulani et al.	2015	Inverse heat transfer problems	Time-step size, noise in measurements and regularization	41	25
Ming et al.	2015	Complexity of water resources system or multi-reservoir system	Maximize energy production	49	26
Das et al.	2016	Multi-robot path planning	Path length, energy optimization	100	27
Elazim and Ali	2016	Power system stabilizers (PSSs) design	Damping oscillation	80	28
Abedi et al.	2017	Numerical optimization in tunnelling	Penetration rate	14	29
Cui et al.	2017	Distance vector-Hop algorithm	Detect Object Problem	07	30
El Aziz and Hassanien	2018	Classification performance in high-dimensional data	Rough sets for feature selection	68	31
Boushaki et al.	2018	Data clustering problems for known data sets	Internal and external clustering quality	36	32

RI: research impact.

the minimal path. It is also called as the computation time in many works of literature. Combining these two parameters formulates more significant prospects of the problem domain. Now there is a need to minimize the path length and that too spending minimum time. To achieve the goal, several optimization techniques<sup>3-6</sup> came into play which is capable of handling these two issues simultaneously. In recent times, several nature-inspired algorithms have been proposed to achieve optimization. These optimization algorithms can be utilized in robot path planning to optimize its parameters like path length, path time, number of turns and many more. There are many works done in the past which applied optimization technique in robot path planning.<sup>7-10</sup>

The cuckoo search (CS) algorithm is one of the widely accepted algorithms for optimization problems, which have been used by many application areas. In this article, we have used the CS algorithm<sup>11</sup> for optimizing the path length and path time for robots path planning. We modified the CS algorithm to perform better than the conventional CS algorithm by applying the tournament selection function.

Here, in this section, a detailed survey of the CS algorithm is discussed and presented.

In this article, proposed a method for the path planning of autonomous mobile robot locally, that is, the search space is unknown or partially known, and the obstacles are static. They used brood parasitic behaviour and levy fight behaviour of cuckoo bird for finding the collision-free path for the mobile robot. The objective function that they used for calculating the optimal path from the start to the goal point is formulated between the robots, target and obstacle. The parameter taken into consideration includes obstacle avoiding behaviour of the robot and the robot's target seeking behaviour. Firstly, the robot moves from the start point to the goal point in a straight line (i.e. shortest between two points). Whenever the robot senses the obstacle in its radar, it implements the CS algorithm. Depending on the objective function value calculated for each nest (i.e. solution) in the swarm, the robot selects the best global solution to avoid obstacles and then proceed towards goal. They claimed better performance and higher efficiency of the proposed method using CS algorithm in optimizing robot navigation by comparing the results with other nature-inspired meta-heuristic algorithms genetic algorithm (GA) and particle swarm optimization (PSO). To analyse the CS algorithm thoroughly. Table 1 is presented in the

article, which gives a brief survey related to CS algorithm. The year-wise development of the algorithm is presented with the parameters of different application areas and its contribution to improving the performance.<sup>33,34</sup>

The article is organized in the following way: First, the necessary working procedure of CS search algorithm is discussed in detail. The next section explains how it is incorporated in robot path planning, after that the next section deals with the details of proposed work, describing the use of tournament function and the CS algorithm. The later section presents the results and discussion followed by the conclusion.

## Cuckoo search

CS algorithm is one of the nature-inspired meta-heuristic algorithms. It is an optimization meta-heuristic search algorithm proposed by Yang and Deb<sup>34</sup> in 2009. Due to its simplistic mathematical processes and capability of solving linear/non-linear problems, it is all over well adopted to solve the optimization-related problems. Some cuckoo's obligated behaviour inspired the working of the CS algorithm.<sup>35</sup> This algorithm is widely accepted in optimization problems like mobile robot navigation, network configuration and many other similar problems. In mobile robot path planning, it reduces the robot's computational time to navigate from one position to another and increases its performance. The hypothesis of the algorithm is constructed on the nature of parasitic organisms. It is well known as obligate brood parasitic behaviour. It is the process where the organism (here the bird cuckoo) relies on some other's nest or at the host to reproduce.

The behaviour of cuckoo's bird inspires the theory of CS algorithm. A group of cuckoo while laying their eggs follows a typical procedure, they lay one egg at a time and put it in some other's nest. Other cuckoo's when found the egg in their nest and identifies that the egg is not her then the host cuckoo destroys the entire egg present in the nest. The host bird discovers the egg laid by a cuckoo with a probability  $P_a[0, 1]$ . All the cuckoo perform this procedure, and at the end, only a few nests having eggs in them survives. The last rule can be calculated by replacing a fraction  $P_a$  of the  $n$  host nests with new nests. The value obtained using the objective function decides the quality of a solution. The ultimate goal is to derive a better solution.

Levy flight method is expended in the CS algorithm for finding solutions. It is a type of random walk which distributes the steps based on the step length. This step length is obtained using the heavy-tailed probability distribution. Levy flight is a random walk in which the increments are obtained using a power law.

$$y = x^{-\beta^t} \quad (1)$$

While working with the CS algorithm, the cuckoo's eggs play an essential and potential role in finding solutions. This algorithm is widely adopted for global optimization

problems. It well balances nicely between the local random walk and the global random walk. This balance between the local and random walk is controlled by a switching parameter of  $P \in [0, 1]$ .<sup>36</sup> Following two equations represents the local and random walk

$$X_i^{t+1} = X_i^t + \alpha s \otimes H(p_a - \varepsilon) \otimes (X_j^t - X_k^t) \quad (2)$$

$$X_i^{t+1} = X_i^t + \alpha(S, \lambda)L \quad (3)$$

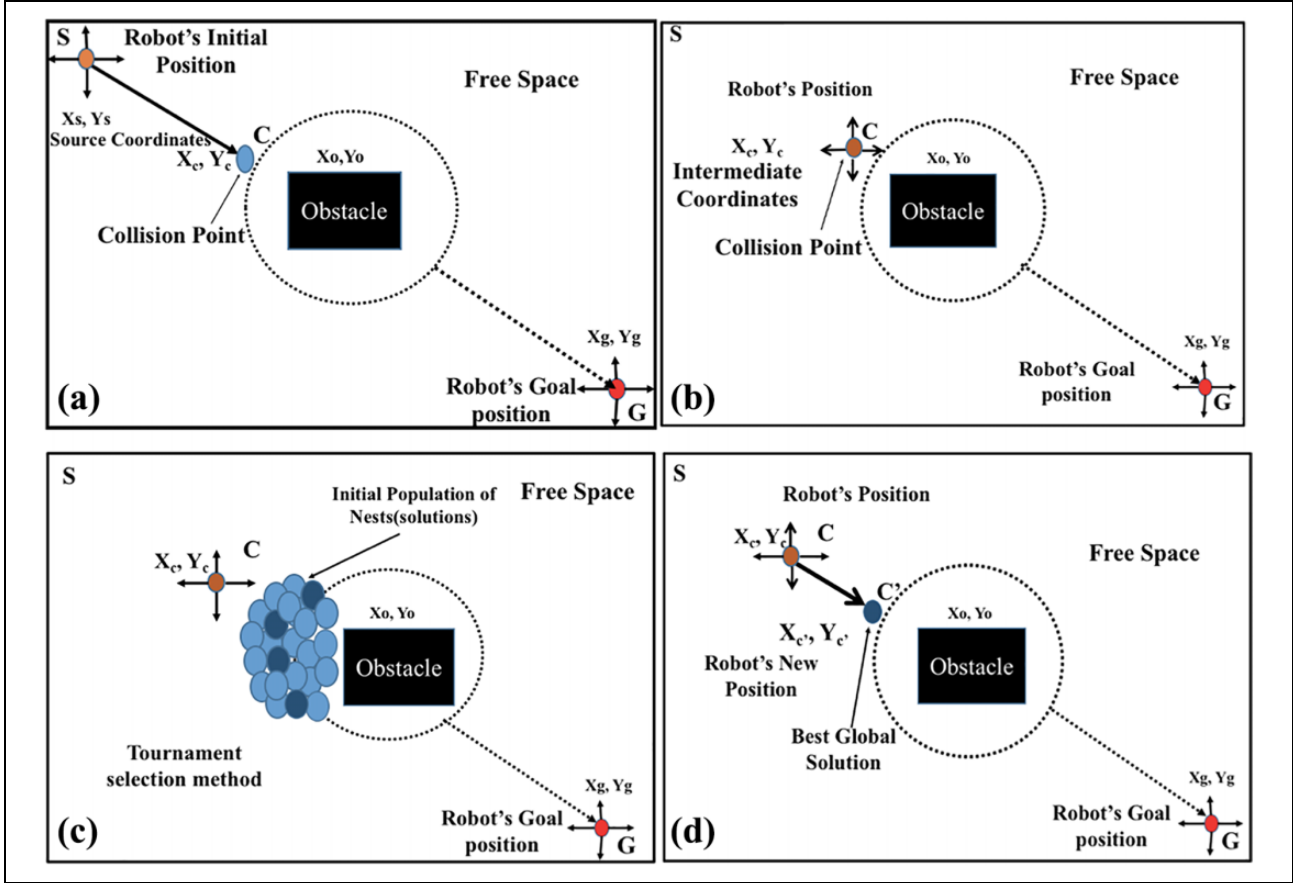
Parameter	Description
$X_i^t, X_k^t$	Initial point selected by random permutation.
$\alpha$	Positive step size scaling factor.
$X^{t+1}$	Succeeding location.
$s$	Size of the step.
$\otimes$	Entry-wise multiplication of two vectors.
$H$	Heavy-side function.
$P_a$	Switching parameter.
$\varepsilon$	Random number from a uniform distribution.
$L(s, \lambda)$	Levy distribution which defines the size of the step.

## Path planning for robots using CS

The navigation problem is primarily transformed into a minimization problem, as most of the service robots navigate in an unknown environment to find the best suitable path to reach the destination. The path is obtained using the objective function. It is calculated by locating the coordinates of the goal and the current obstacle position in the workspace. After formulating the objective function, the CS algorithm is applied to solve the path optimization problem.<sup>37</sup> The best nest selected globally is considered during the entire execution, and the process is repeated in each iteration. After selecting best nest in each round the robot is made to proceed in series utilizing all the selected best nest locations. Moreover, if there are no obstacles found, then the robot approaches towards the destination without any hindrances.

Initially, when a robot starts from its initial position, it continuously moves towards the goal, the problem occurs when the robot encounters an obstacle in its path. When it comes closer to the obstacles, the sensors inbuilt alerts the robot to stop. The path planning algorithm (CS algorithm) is called to find the next move to not collide with any obstacles and gives the optimal way to move forward. Here, the best nest position given by CS is used to do so. Every solution given by CS algorithm is bounded with a numeric value. In goal-seeking, the approach it opts is to measure the two distances from the point, the first distance is from the obstacle which should be maximum, and another one should be from the destination which should be minimum.

The CS algorithm's key point is to select the best nest based on the concept of selecting a nest having a minimum value of their objective function. CS is applied to optimize



**Figure 1.** Different stages showing the working of cuckoo search algorithm that how a robot works and finds optimal path in the given workspace while traversing from source point to goal point.

the path length for the robots. The distance between the robot and the obstacle is the significant criteria for the robot to select the next location in the workspace. It is based on the following conditions:

- Following equation is utilized for finding the distance between the robot and the obstacle. The condition here is that the most considerable distance between them is considered

$$(\text{Dist})_{ROB} = \sqrt{(x_{OB_N} - x_{ROB})^2 + (y_{OB_N} - y_{ROB})^2} \quad (4)$$

- The following equation is utilized to find the distance between the robot and the destination, while the minimum distance between them is considered

$$(\text{Dist})_{RG} = \sqrt{(x_G - x_{ROB})^2 + (y_G - y_{ROB})^2} \quad (5)$$

Satisfying the above two criteria the objective function for the cuckoo's nest can be derived as

$$f(n_i) = c_1 \cdot \frac{1}{\min||n_i - OB_j||} + c_2 \cdot ||n_i - G|| \quad (6)$$

Parameter	Description
$G$	Goal position.
$X_G, Y_G$	Coordinates of the goal.
$N$	Number of obstacles.
$OB_1, OB_2$	Different obstacles.
$X_{OB1}, Y_{OB2}$	Coordinates of the obstacles.
$C_1, C_2$	Controlling parameters

Number of obstacles present are  $OB_d \in (OB_1, OB_2, OB_3, \dots, OB_N)$ .

The controlling parameters of the objective function are selected wisely. The benefit of using the CS algorithm is that the parameters to be adjusted are less than the GA and PSO.<sup>38</sup> Thus, the CS algorithm is more generic to apply to a broader class of optimization problem. A detailed review of the CS algorithm is presented by Parhi et al.<sup>39</sup>

The graphical representation of the working CS algorithm for robot path planning is shown in detail in Figure 1 in four different stages. From Figure 1(a), it is assumed that  $S$  is the robot's start position in the given workspace from where the traversing towards the end point starts.  $X_s$  and  $Y_s$  denote robot's initial coordinates. The path  $SG$  is considered the optimal path as a straight line is the shortest

distance between two points.  $G$  is the goal point where the robot is desired to reach with optimal path length and time. It is represented by the coordinates  $X_g$  and  $Y_g$ . The point,  $C$ , denotes the collision point where the robot's sensing range has detected an obstacle has arrived in its ideal path line. Line  $SC$  is the path in the free space where the robot is moving without any collision.  $X_o$  and  $Y_o$  represent the obstacle's coordinates.

In Figure 1(b), while moving from its start point in the workspace, whenever the robot senses some obstacle in its environment through the sensor present in it, it implements the path planning algorithm; here in this case,  $CS$  algorithm is used to find the next best location for the robot.  $C$  represents the collision point, and the dotted circle denotes the periphery of the obstacle where the traversal is not possible for a robot. This is the point from where the role of path planning algorithm starts for finding the optimal path.

In Figure 1(c), when the robot gets stuck with the obstacle, the  $CS$  algorithm will decide the next best move for the robot according to the values obtained from the objective function for each new solution/random position. The objective function works on the distance criteria between the new random position to the obstacle and the goal/target point. The main objective is to take the new position such that it should be minimum in the distance from the goal and maximum from the obstacle. Initially, the  $CS$  algorithm generates a random number of host nests through levy flight method. Then it selects the current best among them using tournament selection method by comparing the fitness value of  $t$  individuals selected randomly from the total initial population and places the solution over the host nest solution for the next host nest position. This process is done for  $n$  number of times. Afterwards, the algorithm compares the current best solutions/positions according to their objective function values and chooses one position. This position is now the global best solution, and the robot moves a step ahead towards the optimized path.

In Figure 1(d), the position  $C$  is defined to be the collision point, and after resolving the collision or avoiding the obstacle through the modified  $CS$  algorithm, the robot moves towards the global best solution among the current best solutions which is denoted as  $C'$  (new position of the robot). The robot then again continues its usual path if no obstacle is detected and if it counters another obstacle on the way it applies the same approach to overcome it. The  $CS$  algorithm is implemented only if the robot senses some hurdles in its path else it travels towards the goal/target point. Algorithm 1 presents the necessary steps to find the path to be traversed by the robot in the given configuration space. The path obtained thus will be the optimal and collision-free path. The next section details using tournament function to select the next best and the  $CS$  algorithm.

---

**Algorithm 1.** Path planning.

---

**Require:** Time taken

**Ensure:** Traversal

```

1: Initialize  $S_{xy}, G_{xy}$ 
2: Sense for obstacles  $O_{xy}$  in  $E$ 
3: if  $O \in SG$  then
4:   implement  $RCSA$ 
5: end if
6:  $r$  proceeds towards  $G$ 
7: if  $r == G$  then
8:   Stop
9: end if
10: Or Else GOTO step 2

```

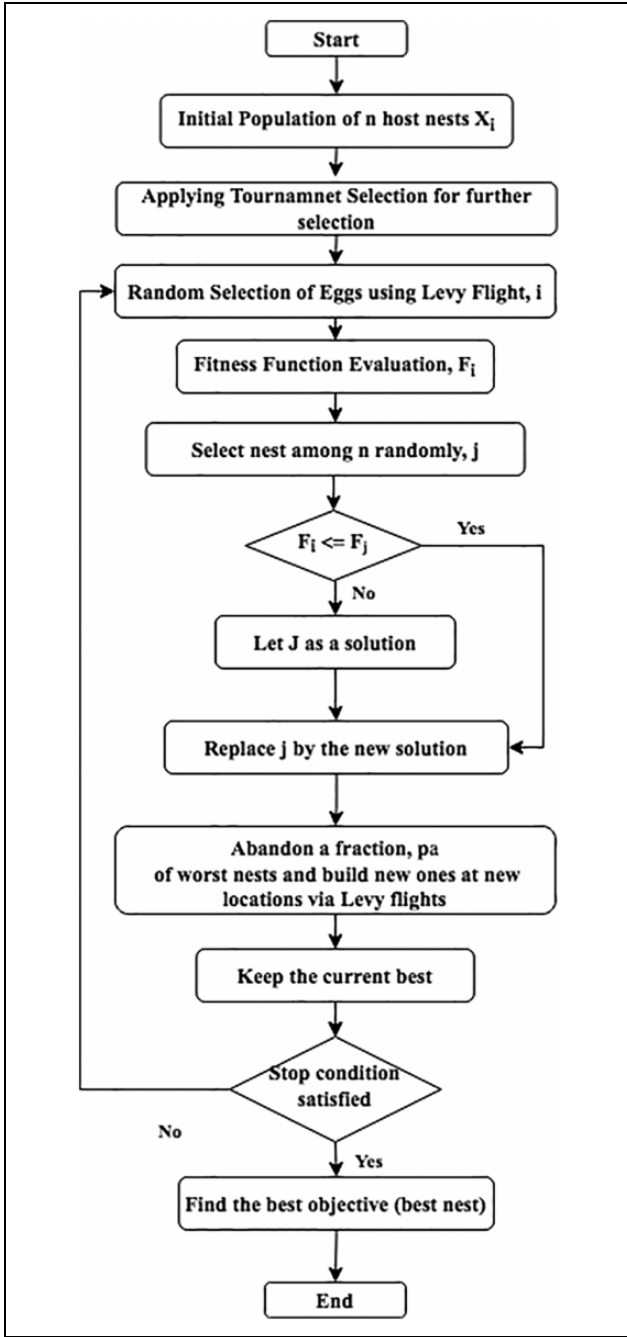
---

## Proposed methodology

In the modified  $CS$  algorithm, we used the tournament selection function to improve the performance. The first selection is made using levy flight to select the nest for placing the cuckoo eggs there and then rest of the selection is made by applying tournament selection function. This helps to optimize the performance of parameters like path length and path time. The  $CS$  algorithm's performance is improved with tournament selection method as it checks for all the possible solution nests with higher fitness value, unlike in levy flight where the nests are selected based on their fitness values chosen randomly. Levy flight is also inefficient for significant problems. This method is mainly used in GAs.<sup>40</sup> In our proposed research work, we are using it with  $CS$  algorithm for robot navigation. Tournament function replaces the criteria of random selection among the solution nests. The random selection of a solution is replaced by the modified  $CS$  algorithm's tournament selection method. The working principle of tournament selection allows all the candidates from the current generation to participate in the tournament, and the selection of  $t$  individuals is made from the initial population. The process continues to select the best individual from the tournament of the next generations. This process is done for  $n$  number of times, and a global best solution is selected as the next solution towards the destination. Figure 2 presents the operational flow of the proposed work, where the selection of different nodes done through tournament function is presented.

### Tournament selection

Exploring and exploiting the given workspace are the two most essential aspects in deciding whether the algorithm's selection process is fair or not. Exploring the traversable space and exploiting the knowledge base (current knowledge base gained by the experience for finding the best solution). The efficient selection of these two factors usually balances the algorithm in finding optimized solution. The tournament selection function is one of the well-known



**Figure 2.** Working flow of the proposed work.

methods for selecting the next best candidates in GA.<sup>41,42</sup> It gives the best results in situations where there is a requirement of choosing a few among many. It selects the best one based on each node's fitness value (called as candidates in terms of GA). The working principle consists of many iterations until the purpose is served. Each round consists of multiple iterations, and in each iteration, multiple candidates participate in the tournament. To select the candidates among all the available nodes, there is another parameter known as selection pressure which is a probabilistic pressure of all the possible candidates based on which

#### Algorithm 2. Tournament selection

- 1: The population  $P(\tau)$  the tournament size  $t \in (1, 2, \dots, N)$
- 2: The population after selection  $P(\tau)$
- 3:  $\text{tournament}(t, J_1, J_2, \dots, J_n)$
- 4: *FOR*  $i \leftarrow 1$  *to*  $N$  *do*
- 5:  $J_i \leftarrow$  best fit individual out of  $t$  randomly picked individuals from  $(J_1, \dots, J_N)$
- 6: *end for*
- 7: *Return*  $(J_1, \dots, J_N)$  and select the best.

each candidate is allowed to participate in the initial stage of the tournament. After competing at the initial stage, the best candidates based on their fitness value qualify for the next round, and the process goes on it, a particular node is selected as the winner of the tournament. Multiple tournaments are organized between the source node and the goal node to reach from source to goal in the robot path planning scenario. The modified approach provides diversity to robot path planning. In the conventional path planning using CS, it uses random selection based on their fitness values. This approach's drawback is that it selects the next based on the current best, which may sometimes lead to entering the local minima and restricts the approach to be utilized in large applications. The use of tournament function overcomes this problem as it has less chance of convergence and makes it the best fit for working in a large workspace.

The first step is to select the initial node while starting from the source point. In the proposed methodology, the first walk around or the selection of the first node is made using the levy flight which is the basic CS algorithm incorporates within, after the first node selection the rest of the selection is made using tournament selection function.<sup>32</sup> Figure 3 explains the working of tournament function with CS algorithm for optimizing path for robots. A flow chart is also shown in Figure 2 to show the modified CS algorithm's working flow for robot path planning. Equation (2) represents the selection based on the levy flight for the first walk around, while equations (3), to (5) represent the method of selecting the further nodes respectively using probability function while deriving path from source to goal. The combination of tournament selection function helps diversify the nature of CS algorithms for many other applications.

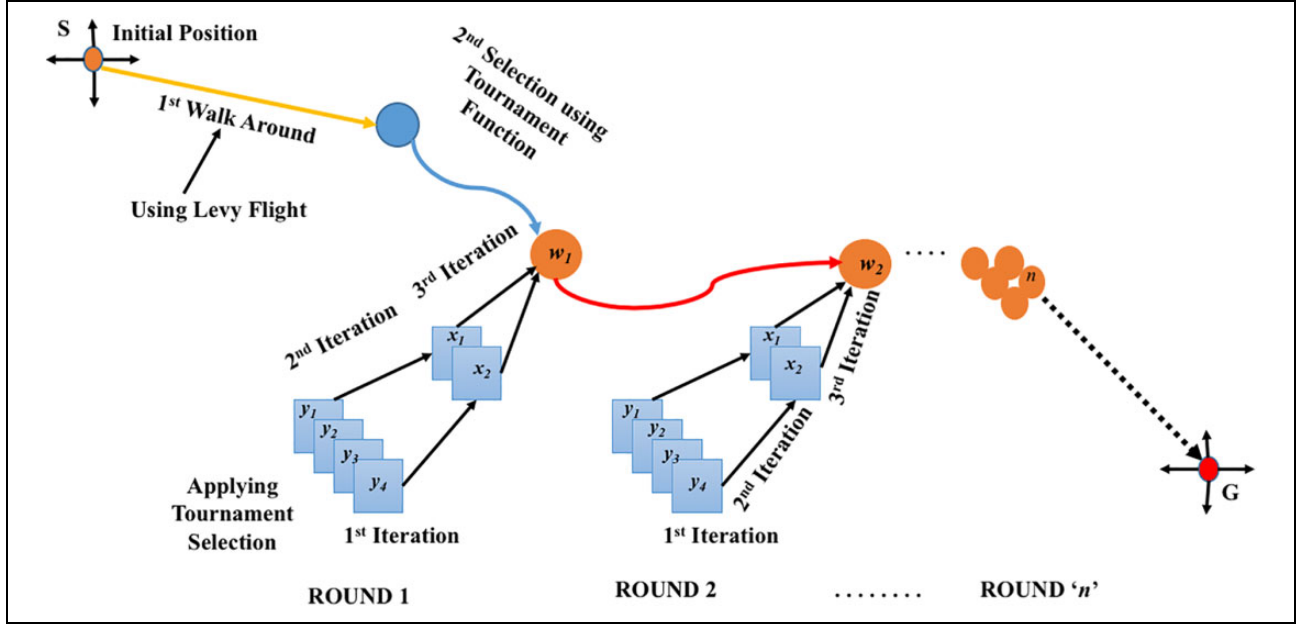
$$\text{LevyFlight} = \frac{\gamma(1 + \beta) \times (\pi + \beta/2)}{\gamma(1 + \beta/2) \times \beta \times 2(\beta - 1)/2} \quad (7)$$

$$\text{InitialSelection} = (p) \quad (8)$$

$$p \times (1 - p), p \times (1 - p)^2 \quad (9)$$

Here in this article, we have taken into account the two most essential parameters, path length and path time, to observe our proposed methodology.





**Figure 3.** The working of proposed methodology of using tournament selection function with cuckoo search algorithm for robot path planning in the static environment for finding an optimal path in the given workspace.

**Path length:** The total distance travelled by the robot in the free space without any collision. Uncertain path planning leads to more number of collisions and thus results in the increased path length. Therefore, path planning should be done to be the shortest path length between the source and goal point. We take the difference between goal coordinate and source coordinate. There are many ways to calculate distance like Euclidean distance, Manhattan distance and Chebychev method. Here we prefer the Euclidean distance

$$f_1 = \sum_{i=1}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (10)$$

**Path time:** The robot's total time in traversing the optimized path formed using a CS algorithm. It is directly proportional to the robot's path length in most cases though there are some complex cases where this rule does not imply, mostly in dense and more complex environments. However, basically in most cases, more the path length, more will be the path time. The optimization is done so that the robot takes a lesser amount of time and path length. This is one of the most critical parameters in the robot's application. **Smoothness:** Two lines which make angle and are the part of the path to be traversed should be able to make robot not to deviate from its path, the value of angle should be such that it should not tend robot to suddenly change path with a little change in the value of the angle. It is the total number of turns taken by the

### Algorithm 3. Modified cuckoo search algorithm.

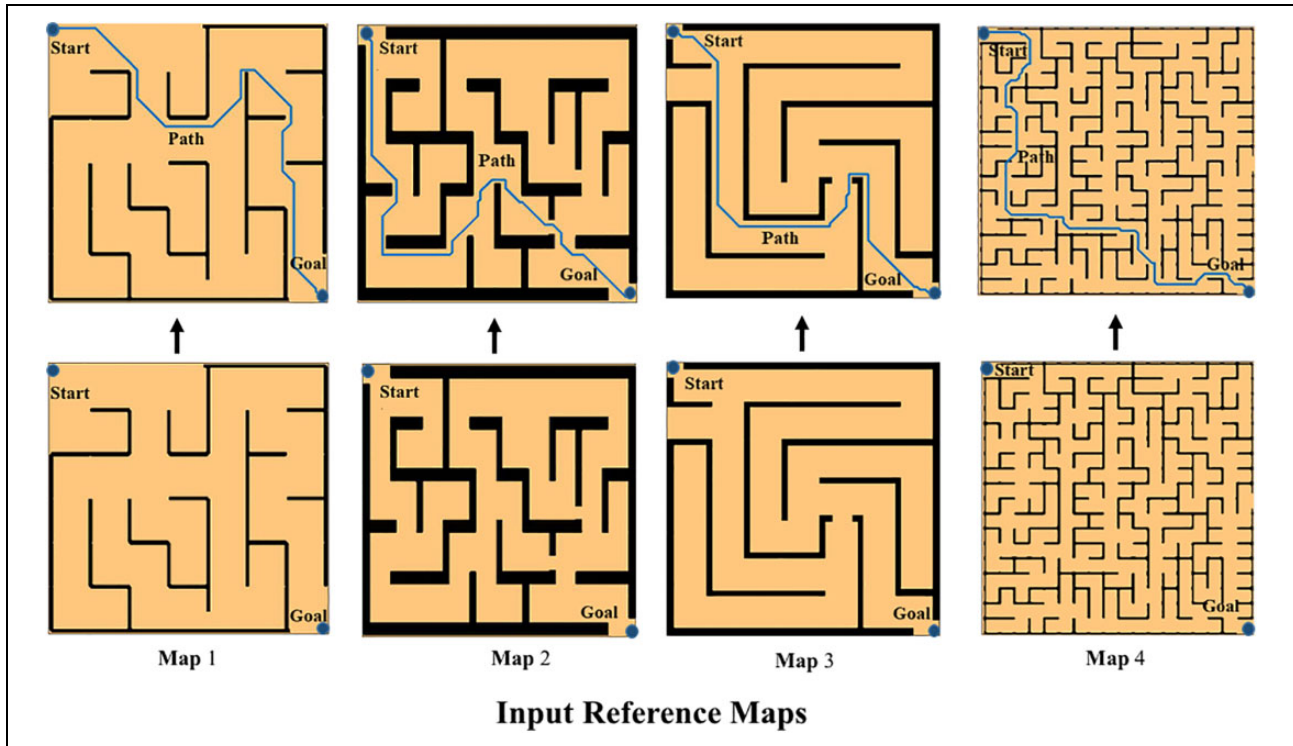
**Require:** Time taken

**Ensure:** Traversal

- 
- Initialise Objective Function  $f(x)$  :*
- 1: Create initial population  $p$  with  $n$  solutions
  - 2: Calculate the fitness value  $f_x$
  - 3: **while**  $t < \text{Max\_iterations}$  or stop criterion **do**
  - 4:   Apply levy flight to get a cuckoo randomly
  - 5:   Calculate fitness value  $f_x$
  - 6:   Apply tournament selection
  - 7:   tournament ( $t, B_1, B_2, \dots, B_n$ )
  - 8:   **for**  $i \leftarrow 1$  to  $n$  **do**
  - 9:      $J'_i \leftarrow \text{bestfit individual out of } t \text{ randomly picked from } B_1, B_2, \dots, B_n$
  - 10:   **end for**
  - 11:   **if**  $F'_j > F_i$  **then**
  - 12:     Replace the current solution with new solution
  - 13:   **end if**
  - 14:   Rank the solutions obtained according to their objective function  $F(X)$
  - 15: **end while**
  - 16: Choose the global best solution from the current best solutions
  - 17: Proceed  $r$  towards new solution  $S'$
- 

robot in reaching towards the goal point. The robot's diagonal movement requires less battery as the path length would be reduced, and thus the total energy robot uses in traversing the path is lowered.  $f_s$  in equation (11) represents the smoothness

$$f_s = \sum_{i=1}^{N-1} \alpha(l_i, l_{i+1}) \quad (11)$$



**Figure 4.** Path obtained after implementing the proposed algorithm using four different reference maps. The top row presents the output of the result while the second row shows the input reference maps, the arrow indicating towards the respective output of reference maps, the blue line in the output map represents the path obtained.

*Success rate:* This parameter accounts for the number of times the proposed methodology provides the correct output. The threshold value could be taken from path length or path time for comparing the results over every iteration here, after applying the CS algorithm on each reference map for a certain number of times.

## Results and discussion

The proposed work uses the concepts of using tournament selection with CS algorithm for finding paths for robots. The conventional procedure is best in finding the next best near the existing best if this procedure continues, which would lead to being trapped in a local minima problem. This is best suited when the workspace is minimal and known. To broaden the horizon, there is a need to skip this trap. This article proposes the tournament selection function to skip the local minima problem while traversing from the source node to the goal node for finding the path for robots.

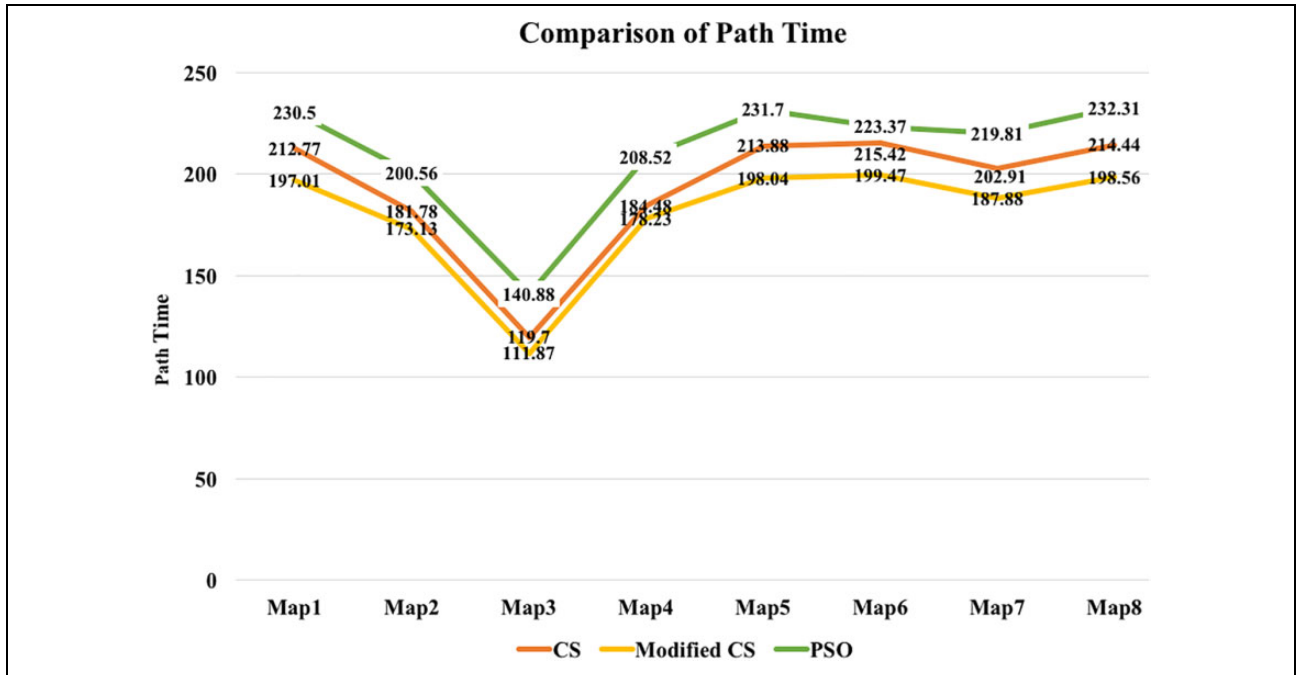
The tournament selection function picks the current best among the general population. Contrary to which the conventional approach leads to the premature convergence restricting it to function in the large workspace. The CS algorithm's performance is improved with tournament selection method as it checks for all the possible solution

neests with higher fitness value. According to the tournament function, several  $t$  size tournaments are organized between  $n$  possible solutions and best among them is selected. This is done for  $n$  number of times, and every time it is being compared with the solution obtained via levy flight and the solution with more fitness value is selected, the current solution gets updated. The solutions are ranked according to the objective function values obtained. Furthermore, the solution with the highest fitness value is considered to be the global best solution. The robot then moves towards the best solution. This is repeated until the robot reaches the target position.

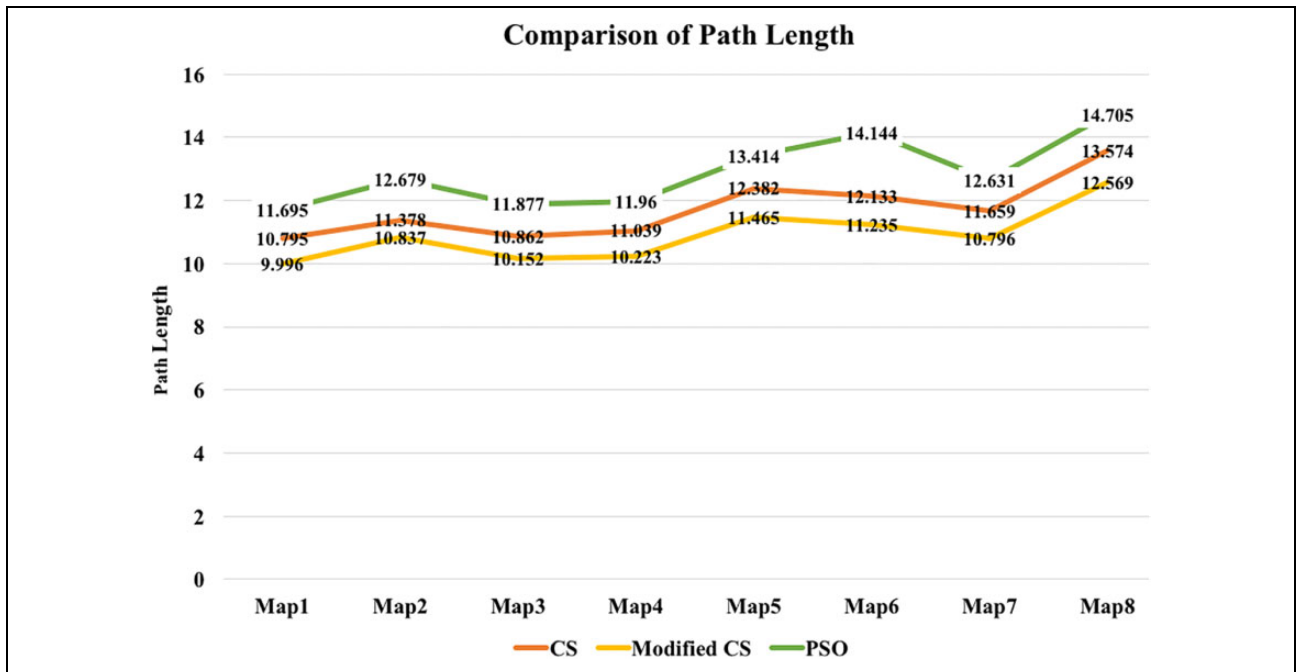
Figure 4 shows the implementation of the proposed approach on four different reference maps. Later on, the experiment is conducted on four other reference maps. The maps consider the diagonal path to be traversed by the robot. In the navigation path, the robot confronts several obstacles to reach the destination from the start point.

The modified CS algorithm's proposed method represents that it performs well in dense maps and provides an optimal path to the robot. The reference maps considered are of  $500 \times 500$  in size. In Figure 4, four different reference maps are presented, which are considered as the input maps where the start and goal points are mentioned. After running the algorithm on this map, the resultant maps with the blue line showing the optimal path from the start node to the goal node





**Figure 5.** Graph showing the comparison of processing time using cuckoo search algorithm and the proposed algorithm using the four different reference maps.



**Figure 6.** Graph showing the comparison of path length using cuckoo search algorithm and the proposed algorithm using the four different reference maps.

are depicted. Figure 5 depicts the performance comparison on the robot's time to navigate the obstacles and reach the target. Figure 6 depicts the performance comparison on the path traversed by both the algorithms. The graph represents the basic CS algorithm, PSO and the modified CS algorithm. The proposed work is

having the edge over the rest two, as shown in the graph. Table 2 presents the comparison of results obtained from the proposed work, the basic CS algorithm and PSO in terms of path length, path time and the success rate which shows that the proposed approach performs better than the CS algorithm.

**Table 2.** Comparison of results obtained, path length and path time on the four different reference maps using cuckoo search algorithm and the proposed approach.

Reference map	Map size	Cuckoo search		Modified cuckoo search			PSO	
		Path length (units)	Path time (s)	Path length (units)	Path time (s)	Success (%)	Path length (units)	Path time (s)
Map 1	500 × 500	10.795	212.77	9.996	197.01	100	11.695	230.50
Map 2	500 × 500	11.378	181.78	10.837	173.13	100	12.679	200.56
Map 3	500 × 500	10.862	119.70	10.152	111.87	100	11.877	140.88
Map 4	500 × 500	11.039	184.48	10.223	178.23	100	11.960	208.52
Map 5	500 × 500	12.382	213.88	11.465	198.04	100	13.414	231.70
Map 6	500 × 500	12.133	215.42	11.235	199.47	100	14.144	223.37
Map 7	500 × 500	11.659	202.91	10.796	187.88	100	12.631	219.81
Map 8	500 × 500	13.574	214.44	12.569	198.56	100	14.705	232.31

PSO: particle swarm optimization.

## Conclusion

This research article presents an approach to find an optimal and collision-free path for robots using modified CS algorithm for path planning. This article has presented a novel approach to replace the robot's random movement based on levy flight with the tournament selection function. The conventional CS algorithm used a random selection of points on the workspace and picked the best among them. This method has a high probability of leaving the nodes that might lead to the best solution. There can be a case where the same point is selected more than once due to the method's random selection criteria. This leads to the local minima problem where the robot gets entangled in the workspace instead of reaching the optimal path. Therefore, we have used tournament selection function in our modified approach. This method randomly picks a few points and conducts a tournament to find the best node among them. This tournament selection is repeated  $n$  number of times with a group of few random individuals. This provides us with the best solution for the robot where it proceeds for the next move. In this way, the local minima problem overcomes the mobile robot path planning in a static environment.

The proposed approach of using tournament selection function with CS algorithm for path selection generates better results while comparing it with the conventional CS algorithm and PSO. Results of path time and path length are presented in the article which performs better in both the cases. The future work could validate the proposed approach for some more parameters like finding smoothness and number of turns while traversing in the workspace. Along with that, there is also a scope to test the proposed approach in the multi-robot scenario. The article has also shown 5%–8% of improvement in the total computation time and path length taken by the robot for traversing from the start node to the goal node between the conventional CS algorithm and the modified CS algorithm, whereas a performance of more than 10% is observed while compared with that of PSO method. The performance improvement

motivates future work to test the proposed approach for multi-robot system or warehouse applications.


## Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## Funding

The author(s) received no financial support for the research, authorship, and/or publication of this article.

## ORCID iD

Kaushlendra Sharma  <https://orcid.org/0000-0002-9140-5788>

## References

1. Kambhampati S and Davis L. Multiresolution path planning for mobile robots. *IEEE J Robot Autom* 1986; 2(3): 135–145.
2. Thorpe C and Matthies L. Path relaxation: path planning for a mobile robot. In: *OCEANS 1984*, Washington, DC, USA, 10–12 September 1984, pp. 576–581. IEEE.
3. Duan H and Qiao P. Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. *Int J Intell Comput Cybern* 2014; 7: 24–37.
4. Hossain MA and Ferdous I. Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. *Robot Auton Syst* 2015; 64: 137–141.
5. Mac TT, Copot C, Tran DT, et al. Heuristic approaches in robot path planning: a survey. *Robot Auton Syst* 2016; 86: 13–28.
6. Pal NS and Sharma S. Robot path planning using swarm intelligence: a survey. *Int J Comput Appl* 2013; 83(12): 5–12.
7. Hosseinienejad S and Dadkhah C. Mobile robot path planning in dynamic environment based on cuckoo optimization algorithm. *Int J Adv Robot Syst* 2019; 16(2): 1729881419839575.
8. Saraswathi M, Murali GB, and Deepak B. Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm. *Procedia Comput Sci* 2018; 133: 510–517.

9. Panda S, Mishra D, and Biswal B. An approach for design optimization of 3R manipulator using adaptive cuckoo search algorithm. *Mech Base Des Struct Mach* 2020; 48(6): 773–798.
10. Song P-C, Pan J-S, and Chu S-C. A parallel compact cuckoo search algorithm for three-dimensional path planning. *Appl Soft Comput* 2020; 94: 106443.
11. Bulmer S and Wessels W. The European Council and its policy-making environment: A cuckoo in the community's nest? In: *The European Council*. Berlin: Springer, 1987, pp. 103–131.
12. Valian E, Mohanna S, and Tavakoli S. Improved cuckoo search algorithm for feedforward neural network training. *Int J Artif Intell Appl* 2011; 2(3): 36–43.
13. Layeb A. A novel quantum inspired cuckoo search for knapsack problems. *Int J Bio-Inspir Comput* 2011; 3(5): 297–305.
14. Subotic M, Tuba M, Bacanin N, et al. Parallelized cuckoo search algorithm for unconstrained optimization. In: *Proceedings of the fifth WSEAS congress on applied computing conference, and Proceedings of the first international conference on biologically inspired computation*, 2 May 2012, pp. 151–156. Stevens Point Wisconsin United States: World Scientific and Engineering Academy and Society (WSEAS).
15. Yang X-S, Deb S, Karamanoglu M, et al. Cuckoo search for business optimization applications. In: *2012 National conference on computing and communication systems*, Durgapur, India, 21–22 November 2012, pp. 1–5. IEEE.
16. Yang X-S and Deb S. Multiobjective cuckoo search for design optimization. *Comput Oper Res* 2013; 40(6): 1616–1624.
17. Yildiz AR. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *Int J Adv Manuf Technol* 2013; 64(1–4): 55–61.
18. Basu M and Chowdhury A. Cuckoo search algorithm for economic dispatch. *Energy* 2013; 60: 99–108.
19. Ouaarab A, Ahiod B, and Yang X-S. Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Comput Appl* 2014; 24(7–8): 1659–1669.
20. Mohanty PK and Parhi DR. Cuckoo search algorithm for the mobile robot navigation. In: *International conference on swarm, evolutionary, and memetic computing*, Chennai, India, 19–21 December, 2013, pp. 527–536. Springer.
21. Bhandari AK, Singh VK, Kumar A, et al. Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. *Expert Syst Appl* 2014; 41(7): 3538–3560.
22. Marichelvam M, Prabakaran T, and Yang X-S. Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Appl Soft Comput* 2014; 19: 93–101.
23. Goyal S and Patterh MS. Wireless sensor network localization based on cuckoo search algorithm. *Wirel Pers Commun* 2014; 79(1): 223–234.
24. Nguyen TT, Vo DN, and Truong AV. Cuckoo search algorithm for short-term hydrothermal scheduling. *Appl Energy* 2014; 132: 276–287.
25. Mulani K, Talukdar P, Das A, et al. Performance analysis and feasibility study of ant colony optimization, particle swarm optimization and cuckoo search algorithms for inverse heat transfer problems. *Int J Heat Mass Transf* 2015; 89: 359–378.
26. Ming B, Chang J-X, Huang Q, et al. Optimal operation of multi-reservoir system based-on cuckoo search algorithm. *Water Resour Manag* 2015; 29(15): 5671–5687.
27. Das PK, Behera HS, and Panigrahi BK. A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. *Swarm Evol Comput* 2016; 28: 14–28.
28. Elazim SA and Ali E. Optimal power system stabilizers design via cuckoo search algorithm. *Int J Electr Power Energy Syst* 2016; 75: 99–107.
29. Abedi Firouzjaee H, Kordestani JK, and Meybodi MR. Cuckoo search with composite flight operator for numerical optimization problems and its application in tunnelling. *Eng Optim* 2017; 49(4): 597–616.
30. Cui Z, Sun B, Wang G, et al. A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. *J Parallel Distrib Comput* 2017; 103: 42–52.
31. El Aziz MA and Hassanien AE. Modified cuckoo search algorithm with rough sets for feature selection. *Neural Comput Appl* 2018; 29(4): 925–934.
32. Boushaki SI, Kamel N, and Bendjeghaba O. A new quantum chaotic cuckoo search algorithm for data clustering. *Expert Syst Appl* 2018; 96: 358–372.
33. Yang X-S and Deb S. Cuckoo search: recent advances and applications. *Neural Comput Appl* 2014; 24(1): 169–174.
34. Yang X-S and Deb S. Cuckoo search: state-of-the-art and opportunities. In: *2017 IEEE fourth international conference on soft computing & machine intelligence (ISCMI)*, Port Louis, Mauritius, 23–24 November 2017, pp. 55–59. IEEE.
35. Dey N. *Applications of cuckoo search algorithm and its variants*. London: Springer Nature, 2020.
36. Mareli M and Twala B. An adaptive cuckoo search algorithm for optimisation. *Appl Comput Inform* 2018; 14(2): 107–115.
37. Fister I, Yang X-S, and Fister D. Cuckoo search: a brief literature review. In: *Cuckoo search and firefly algorithm*. Cham: Springer, 2014, pp. 49–62.
38. Sharma K and Doriya R. Path planning for robots: an elucidating draft. *Int J Intell Robot Appl* 2020; 4: 294–307.
39. Mohanty PK and Parhi DR. Optimal path planning for a mobile robot using cuckoo search algorithm. *J Exp Theor Artif Intell* 2016; 28(1–2): 35–52.
40. Liu C and Kroll A. Performance impact of mutation operators of a subpopulation-based genetic algorithm for multi-robot task allocation problems. *SpringerPlus* 2016; 5(1): 1361.
41. Yang J and Soh CK. Structural optimization by genetic algorithms with tournament selection. *J Comput Civil Eng* 1997; 11(3): 195–200.
42. Tian W, Meng W, and Sun M. Mapping route optimization in warehousing environment based on improved genetic algorithm. In: *2018 IEEE international conference on real-time computing and robotics (RCAR)*, Kandima, Maldives, 1–5 August 2018, pp. 343–347. IEEE.