

# A HBMO-based batch beacon adjustment for improving the Fast-RRT

Heru Suwoyo<sup>1</sup>, Yingzhong Tian<sup>2</sup>, Andi Adriansyah<sup>1</sup>, Julpri Andika<sup>1</sup>

<sup>1</sup>Department of Electrical Engineering, Faculty of Engineering, Universitas Mercu Buana, Jakarta, Indonesia

<sup>2</sup>School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China

## Article Info

### Article history:

Received Apr 24, 2024

Revised Oct 10, 2024

Accepted Oct 30, 2024

### Keywords:

Fast-RRT

Global path planning

HBMO

Mobile robot

Path planning based-hybrid algorithm

## ABSTRACT

Fast-RRT improves on the original rapidly-exploring random trees (RRT) by incorporating two main stages: improved-RRT and fast-optimal. The improved-RRT stage enhances the search process through fast-sampling and random steering, while the fast-optimal stage optimizes the path using fusion and path arrangement. However, path fusion can only be optimal when the newly found path is unique and different from previous paths. This uniqueness rarely occurs in cases with narrow corridors, so path fusion only provides suboptimal conditions. To address this, the study explores using honey bee mating optimization (HBMO) to optimize or replace the fusion stage. HBMO helps determine new beacon coordinates, which are nodes between the start and goal points along the path, through a batch beacon adjustment approach. The results show that integrating HBMO into Fast-RRT improves its optimality, with a 21.85% reduction in path cost and a 5.22% decrease in completion time across environments with varying difficulty levels. This hybrid algorithm outperforms previous methods in terms of both path optimality and convergence rate, demonstrating its effectiveness in enhancing Fast-RRT's performance.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Julpri Andika

Department of Electrical Engineering, Faculty of Engineering, Universitas Mercu Buana  
Jakarta, Indonesia

Email: [julpri.andika@mercubuana.ac.id](mailto:julpri.andika@mercubuana.ac.id)

## 1. INTRODUCTION

Not only behavior-based movement [1]-[3], simultaneous localization and mapping (SLAM) [4], [5] and path tracking [6]-[8], path planning is also an issue that must be considered to make robots able to navigate autonomously in unknown environment [4], [9]. Searching and sampling-based path planning algorithms serve as two major approaches in robotics and artificial intelligence, each having distinct characteristics. Search-based algorithms such as A\* [10] and Dijkstra [11], [12] prioritize the search for optimal paths through systematic exploration of the search space, ensuring completeness and optimality. They are efficient in smaller, static environments where the right solution is paramount. In contrast, sampling-based methods such as rapidly-exploring random trees (RRT) [13] and probabilistic roadmaps (PRM) [14] focus on exploring feasible regions of space through random sampling, making them particularly adept in high-dimensional and dynamic environments. While it does not guarantee optimality, it offers scalability and adaptability, making it suitable for scenarios with complex obstacles or constantly changing terrain. The choice between the two depends on factors such as environmental dynamics, computing resources, and the desired balance between optimality and efficiency. Many researchers and practitioners utilize sampling-based path planning methods to increase optimality while taking advantage of the speed

available in finding solutions. Sampling-based algorithms, such as RRT and its variants, prioritize efficiency by exploring the configuration space through random sampling and incremental tree growth.

Although RRT may not guarantee an optimal solution in its basic form, researchers have developed techniques to increase its optimality without reducing its speed. For example, algorithms such as RRT-Connect [15]-[17]. RRT-Connect carries a two-way approach in finding paths between start points and end points in the search space. By initializing an exploration tree from a starting point, RRT-Connect iteratively performs a random expansion, adding new points connected to nearby points in the tree. This process continues until the path reaches the end point, then the reverse process is carried out from the end point to the start point. By combining the two trees from opposite directions, RRT-Connect creates sufficient connection paths between the start and end points in the search space.

Although it does not guarantee an optimal path, this approach is efficient in exploring the search space and often provides solutions in a short time. In response to these limitations, rapidly-exploring random trees star (RRT\*) [13], [18] was introduced, improving the algorithm with a different approach. RRT\* combines a random exploration approach with a node evaluation and reconnection process to improve the quality of the resulting paths.

Additionally, RRT\* dynamically updates existing paths by considering the total cost from the starting point to each node in the exploration tree. Thus, RRT\* enables the search for more efficient and potential paths to reach optimal solutions in various motion planning problems. However, the non-specific direction of exploration means that RRT\* has the potential to conduct sampling in places that have actually never been touched or even have no significance for excessive exploration. This makes the convergence speed slow and needs attention. Referring to this limitation, RRT\* was improved with a hybrid approach that utilizes the artificial potential field method [19]-[21]. The use of APF in RRT\* is intended to produce more targeted exploration within a given environment [20], [21]. The integration of the two aims to accelerate convergence towards a solution by reducing the number of iterations required, as well as reducing execution time to reach the optimal path. The APF algorithm is known for its simplicity and powerful mathematical analysis, despite its limitation to state spaces of up to five dimensions due to its inability to operate in local minima environments. Apart from that, Quick-RRT\* has also been introduced with the motivation of an exploration method with good convergence speed [9], [22]. Quick-RRT\* integrates the concept of triangle inequality to speed up the process of determining nodes and connections in exploration trees. By using triangle inequalities, Quick-RRT\* can reduce the number of distance evaluations between existing nodes, resulting in faster path finding. This approach optimizes search steps by prioritizing expansion to nodes that have the greatest probability of directing a path to the destination point.

Therefore, Quick-RRT\* significantly speeds up the motion planning process, making it an attractive option in situations where speed of execution is an important factor. Furthermore, based on the benefits of both APF- and Quick-based methods, a hybrid approach such as PQ-RRT\* is proposed, combining the strengths of each to achieve improved performance in planning tasks [23], [24]. Although Quick-RRT\* can speed up the motion planning process by using the concept of triangle inequality to prune the evaluation of distances between vertices, aggressive pruning can result in suboptimal paths [25]. This process can eliminate nodes that are important for creating a better path, and result in solutions that are further from optimal. In addition, the sensitivity to parameters and the difficulty of adjustment also makes the implementation complexity also increase. Therefore, other methods are introduced with a focus on maintaining speed and optimality as ideally as possible.

Different from the previous one, this method was built by considering the RRT as a base. The scientific reason is the desire not to rely on repeated search processes as in RRT\*. This method is called Fast-RRT [26], [27]. Although there is no reconnection process in this method, sampling is adopted from the improved-RRT. The improvement in question is the addition of fast-sampling and random-steering processes. Fast-sampling is a way of generating random nodes without repetition in the touched area. Meanwhile, random steering is applied to control the generation of random nodes in the direction towards the target point. One advantage of improved RRT in fast-RRT is that it can speed up exploration even in environments with narrow passageways thanks to its fast-sampling and random steering. Fast-sampling, on the other hand, removes the chance that the first path found is not the best one because it restricts sampling to unexplored areas. Furthermore,  $X_{new}$  obtained will have very limited connection options to other nodes in  $T$  in the absence of wiring such as RRT\*. As a result, the newly created path will typically resemble the previous  $T_{opt}$ . This has an impact on the path fusion process, which ought to be at its best when the newly formed path differs from the old one as well. Therefore, the RRT\* wiring process is used in this study. In addition, Fast-RRT also introduces the path-optimal stage which contains a sequential process between the fusion and tuning paths as an effort to maintain the optimality of the path formed. Thus, Fast-RRT has better performance when compared to RRT or RRT\* in terms of convergence speed and optimality.

However, in reality Fusion will only produce new paths that are less than optimal and slow down planning time. Therefore, there needs to be development of Fast-RRT with objectivity to replace the fusion stage. In this research, the honey-bee mating optimization (HBMO) [28]-[32] method was involved. Its involvement is to connect random nodes around the nodes on the path resulting from the initial Fast-RRT stage with reference to the shortest distance. As an optimization method, HBMO can be applied in determining the optimal solution, both minimum and maximum, by considering the objective function or a single numerical quantity. So, this is enough to prove that HBMO has a high level of feasibility to represent the expected solution. Although there are many variants of the method, such as simulated annealing (SA) [31], grey wolf optimization (GWO) [33]-[36], artificial bee colony (ABC) [29], [37], [38], particle swarm optimization [39]-[42], and genetic algorithm (GA) [29], [35], [36], HBMO has capabilities that cannot be by other methods.

To maximize the optimality of the results, HBMO is essentially used to control the position of nodes other than the goal and start contained in  $T_{opt}$ . Updating these positions are used to obtain the optimal path,  $T_{opt}$ , which is  $T_{init}$ .  $T_{opt}$  is made up of beacon nodes that link  $x_{init}$  and  $x_{goal}$ . It is evident that this path has fewer beacon nodes than all the nodes in  $T_{init}$ , except for  $x_{init}$  and  $x_{goal}$ , since it creates a collision-free direct connection between nodes and the farthest nodes. Then, in the HBMO optimization process, the number of beacon nodes,  $N$ , is considered as a decision variable. Due to their small number, HBMOs should optimize their locations by shifting them to pertinent areas. For every beacon node, this displacement is constrained by the radius  $r$ . The only other requirements for this decision variable are to maintain a connection without running into any environmental obstacles. Furthermore, the fitness function used is the path cost, which determines the total connection distance from  $x_{start} - x_{goal}$  through the beacon nodes, considering that the path from the beacon node connection represents a candidate solution. Therefore, it is evident that this kind of optimization refers to minimizing the search space within a time constraint to generate optimal paths.

The HBMO generation is controlled based on the number of samples left over after reaching the designated maximum sampling based on this time restriction. This kind of HBMO deployment pattern enables the beacon node to move to a new location and possibly establish a direct connection with another beacon node that is the farthest away. At the conclusion of the process, path optimization is done once more before the final path is decided to break the cycle of finding the best path. Referring to a search cycle like this, this method was later named HBMO-optimized Fast-RRT\* with the latest form being the creation of Fast-RRT which adds a rewiring process, and the implementation of HBMO after path optimization has been carried out. The performance of the proposed method for solving a global path planning are compared to its predecessor, namely RRT\* and Fast-RRT, in terms of optimality and convergence rate. And according to the comparative result, the proposed method shows a significant improvement.

The rest of this paper is organized as follows: the material and methods are presented in section two. It covers the problem statement, Fast-RRT, and HBMO; the proposed method is presented in section three including the pseudocode and flowchart; the result and discussion is discussed in section four; and section five presents the conclusion.

## 2. METHOD

In this section, the materials and methods involved in this research are presented. It includes the problem statement, brief review on Fast-RRT, and honey bee mating optimization. They are presented to provide more clarity on the methods that have been developed in the next section.

### 2.1. Problem statement

Let  $X \in \mathbb{R}^n$  is representation of state space for a path planning problem, with  $n \in N$  is space dimension, thus  $X = \{X_{obs}, X_{free}\}$  is state space with  $X_{obs} \in X$  refers to obstacle coordinates and  $X_{free} \in X$  refers to the free space. Moreover, if the start node  $x_{init} \in X_{free}$  and goal node  $x_{goal} \in X_{free}$  are given, then referring to  $X_{obs}$ , the path planning algorithm has to find the ideal path from-to those nodes, denoted as  $\sigma = [0, T] \rightarrow X_{free}$  with  $\sigma(0) = x_{init}$  and  $\sigma(T) = x_{goal}$  where  $X_{goal} = \{x \in X | x - x_{goal} | < r\}$  for  $r$  is radius around  $x_{goal}$ .

### 2.2. Fast-RRT

In general, Fast-RRT consists of two main steps in solving global path planning problems, namely Improved-RRT and Fast-Optimal. Respectively, they are used to find feasible paths and to optimize paths by combining current and previous paths. Both are executed sequentially to determine the best path. This can be seen in Algorithm 1.

**Algorithm 1. Fast-RRT**

```

Input:  $x_{init}, X_{goal}, Map$ 
Output: A path  $T$  connecting  $x_{init}$  and  $x_{goal}$ 
for  $i = 1:N$ 
 $T_{init} \leftarrow improvedRRT(x_{init}, X_{goal}, Map)$ 
if  $T_{init}$  is found
 $T_{opt} \leftarrow fastoptimal(T_{init}, T_{opt})$ 
endif
endfor

```

Both the improved-RRT and fast-optimal stages each have two sub-step descriptions which make them different from their predecessor, namely RRT. In improved-RRT there is fast sampling which is designed not to repeat sampling in areas that have already been explored, and random steering which is used to change the relationship between two nodes if there is a collision with an obstacle in the environment. The representative algorithm for improved-RRT can be seen in Algorithm 2.

**Algorithm 2. Improved-RRT**

```

Input:  $x_{init}, X_{goal}, Map$ 
Output: A path  $T$  connecting  $x_{init}$  and  $x_{goal}$ 
 $x_{rand} \leftarrow fastsampling(Map)$ 
 $x_{near} \leftarrow Near(T, x_{rand})$ 
 $x_{new} \leftarrow randomSteer(x_{near}, x_{rand})$ 
 $E_i \leftarrow Edge(x_{near}, x_{new})$ 
if  $CollisionFree(Map, E_i)$ 
 $T \leftarrow addNode(x_{new}, E_i)$ 
if  $x_{new} \in X_{goal}$ 
 $T_{init} \leftarrow getPath(T)$ 
endif
endif
endfor

```

where,  $Edge(\cdot)$  is a function to get a connection node to node,  $addNode(\cdot)$  appends the node to trees  $T$ , and  $T_{init}$  is a feasible path from  $x_{init}$  to  $x_{goal}$  obtained by tracing all nodes based their own parent. Whereas  $fastSampling(\cdot)$  is stated as shown in Algorithm 3.

**Algorithm 3. Fast sampling**

```

Input:  $Map$ 
Output:  $x_{rand}$ 
 $x_{rand} \leftarrow uniformSampling(Map)$ 
while  $x_{rand} \in X_{explored}$ 
 $x_{rand} \leftarrow uniformSampling(Map)$ 
endwhile

```

where  $x_{rand}$  in  $X_{explored}$  is detected by measuring each node on  $T$  to  $x_{rand}$ . It is assumed to be detected if the distance is less than a predefined radius. Meanwhile,  $randomSteer(\cdot)$  is done by executing Algorithm 4.

**Algorithm 4. Random steering**

```

Input:  $Map, T, x_{rand}$ 
Output:  $x_{new}$ 
 $x_{near} \leftarrow Near(T, x_{rand})$ 
 $E_i \leftarrow Edge(x_{near}, x_{near})$ 
if  $CollisionFree(E_i, Map)$ 
return  $x_{new}$ 
else
 $\theta \leftarrow rand(2\pi)$ 
 $x_{new} \leftarrow Expand(x_{near}, \theta)$ 
endif

```

where  $\theta$  refers to the steering angle of  $x_{near}$ - $x_{new}$  connection, which is randomly generated in range of  $[0, 2\pi]$  if  $x_{new}$  collides to the obstacle. And  $expand(\cdot)$  is a function used to refine  $x_{new}$  according to  $x_{near}$  and  $\theta$ . Furthermore, in Fast-Optimal there are also two sub steps, namely path fusion and path tuning. As the name suggests, fusion involves two previously obtained paths with a newly obtained path, which is then combined into a path with better quality based on the intersection point. For the record, this intersection point is the intersection point of the two processed paths, and is obtained by finding the average position based on two close points. The process of path fusion can be seen in Algorithm 5.

**Algorithm 5. Path fusion**

```

Input:  $T_{init}, T_{opt}$ 
Output:  $T_{opt}$ 
for each point in  $T_{opt}$ 
for each point  $T_{init}$ 
if  $\|point_{T_{opt}}, point_{T_{init}}\| < Thr$ 
 $intersection_{idx} \leftarrow append(idxPoint_{T_{opt}}, idxPoint_{T_{init}})$ 
endif
endfor
endfor
 $CostStartIntA \leftarrow \|T_{init}(start):T_{init}(intersection_{idx})\|$ 
 $CostStartIntB \leftarrow \|T_{opt}(start):T_{init}(intersection_{idx})\|$ 
 $CostIntGoalA \leftarrow \|T_{init}(intersection_{idx}):T_{init}(goal)\|$ 
 $CostIntGoalB \leftarrow \|T_{opt}(intersection_{idx}):T_{opt}(goal)\|$ 
if  $CostStartIntA < CostStartIntB$ 
SubpathA =  $T_{init}(start):T_{init}(intersection_{idx})$ 
else
SubpathA =  $T_{opt}(start):T_{init}(intersection_{idx})$ 
endif
if  $CostIntGoalA < CostIntGoalB$ 
SubpathB =  $T_{init}(intersection_{idx}):T_{init}(goal)$ 
else
SubpathB =  $T_{opt}(intersection_{idx}):T_{opt}(goal)$ 
endif
 $T_{opt} = [SubpathA, SubpathB]$ 

```

After the fusion process is complete, the resulting path is then tuned using the path fine tuning process. No different from the optimization process in general, this path fine tuning applies triangular inequality to shorten the path. With this process the initial point will try to be connected to the next point. When there is no collision with an obstacle, the starting point is connected to a further point, and so on. However, the point connection is invalid if a collision occurs and the Subpath is the connection between the initial point and the last point which is still free of obstacles, and the initial point is determined again. The starting point for the second Subpath is the point connected to the starting point on the Subpath that has been formed. Next, the process repeats, until the target point is successfully connected to the last Subpath. Finally, the path generated by path fine tuning combines these Subpaths.

**2.3. Honey bee mating optimization (HBMO)**

As a potential metaheuristic algorithm, the HBMO algorithm mimics the drone, brood, and queen. Genotype, speed, energy, and spermatheca with a certain capacity are characteristics of each queen. The drone's sperm store, known as the spermatheca, is created during mating with the queen. Consequently, speed and energy are initialized prior to each mating trip for queens with a specific spermatheca size. The drone sperm is kept in the queen's spermatheca following a successful mating attempt. The remaining genes from the queen's genome are completed later in the breeding process, and some drone genes are copied into the parent's genotype to generate the brood. The queen chooses a drone at random from the issue choice space, and mating flights are mapped into the HBMO algorithm. The queen's spermatheca contains the genome of every drone that mates successfully. Additionally, in each iteration of the HBMO algorithm, all drones left after the mating flight were destroyed to simulate drone mortality at the end of the breeding season. Model worker bees use heuristic functions to try to enhance the quality of their brood. As a potential genetic answer, the HBMO algorithm mimics the drone, brood, and queen. Workers' brood care is translated into an algorithm that uses heuristic functions to improve brood. The drone's sperm store, known as the spermatheca, is created during mating with the queen. Consequently, speed and energy are initialized prior to each mating trip for queens with a specific spermatheca size. The drone sperm is kept in the queen's spermatheca following a successful mating attempt. The remaining genes from the queen's genome are completed later in the breeding process, and some drone genes are copied into the parent's genotype to generate the brood. The queen chooses a drone at random from the issue choice space, and mating flights are mapped into the HBMO algorithm. The queen's spermatheca contains the genome of every drone that mates successfully. Next, a drone (solution) is chosen at random from the decision space to mate with the queen using mating flight. Each chosen drone's genome is kept in the spermatheca of the queen. Each chosen drone's genome is kept in the spermatheca of the queen. By using crossover and mutation operators between the queen and the solution kept in the queen's spermatheca, diploid broodstock are produced. By using crossover and mutation operators between the queen and the solution kept in the queen's spermatheca, diploid broodstock are produced. To enhance the quality of the brood, the heuristic function that simulates worker bees is used. The HBMO algorithm's flow diagram is shown in Figure 1.

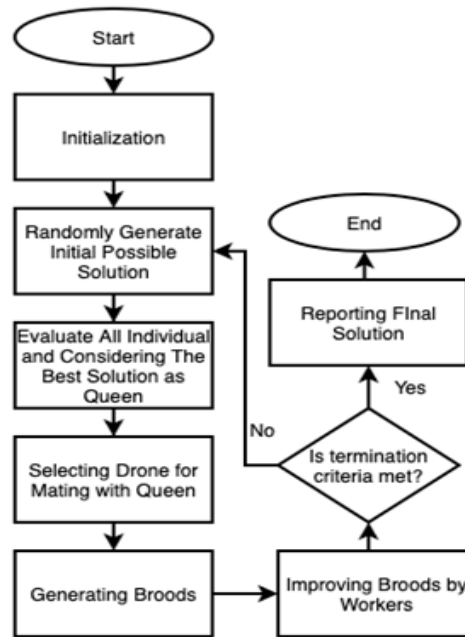


Figure 1. Flowchart of HBMO

### 3. PROPOSED METHOD

The fast-sampling and random steering found in improved-RRT in Fast-RRT have the advantage of being able to speed up the exploration process, even in environments with narrow passageways. However, fast-sampling, which limits sampling to areas that have not yet been explored, actually eliminates the possibility that the initial path obtained is less than optimal. In addition, without wiring such as RRT\*,  $x_{new}$  obtained will have very few connection options to other nodes in  $T$ . So, the new path that is formed will tend to have similarities to the previous  $T_{opt}$ . This affects the path fusion process which should be optimal when the new path formed also has variations from the previous one. Thus, in this research the wiring process on the RRT\* is applied. In the proposed method, the implementation is carried out after edge  $E_i$  is obtained.

After  $x_{new}$  is added to tree  $T$  and its parent is assigned, the exploration process is repeated until the termination criterion is met, namely when  $x_{new} \in X_{goal}$ . This process then produces results in the form of the initial path  $T_{init}$ . Instead of applying Fast Optimal, in this research the application of the metaheuristic Algorithm 6, HBMO, was carried out. This implementation is based on the shortcomings of path fusion found in Fast-RRT, namely that it has the potential to not be executed because, newly varied paths are rarely obtained by simply increasing sampling only. Another reason is that it takes quite a long time to execute the path fusion process. Therefore, this is contrary to the initial motivation of fast sampling, which was held with the aim of increasing the convergence rate of Fast-RRT.

Basically, HBMO is carried out to regulate the position of nodes other than the goal and start contained in  $T_{opt}$  as an effort to increase the optimality of results. The best bee candidate, which successfully shows the shortest path value, is assumed to be the queen in HBMO. This queen will then be mated with the drone. The bees that are not selected as queens are assumed to be drone candidates. By utilizing roulette wheel selection, drone candidates will be selected. Technically, this selection refers to the results of the previous fitness calculation. The total of the overall fitness becomes the divisor, for each drone fitness. Here, the representation value will be obtained which is then used as a reference in making the Cumulative Probability Distribution. Then by creating a cumulative probability array, roulette wheel selection is simulated. This cumulative probability array is a reference where each individual occupies a wheel segment that is proportional to its selection probability. Assuming that the roulette wheel selector is a random number from 0-1, then clearly the coordinates of the node that produces a shorter distance than the others have the potential to be selected as a drone. After the drone is selected, the simulation of the mating process between the drone and the queen is carried out. In this simulation, the principles of crossover and mutation are carried out. Both processes involve changing the coordinate data into binary form. In this study, the coordinate value was changed to 8-bit binary. In the crossover process, the good bits on the queen and drone are exchanged using the single-point crossover method. The crossover point divides the chromosome into two parts on each

drone and queen. And the exchange is done on the bit collection after the crossover point. While in mutation, which is done after the crossover is complete, the study applies bit flipping to the chromosomes of both drones and queens. The results of the crossover and mutation processes are then called brood in Figure 1. Next, both of them which represent the characters of the drone and queen are tested based on their fitness values. This is to ensure that between the two there is one that is better than the queen. If not, the queen will be the same and the drone selection is repeated. And so on, until a better brood is produced, which then replaces the drone and queen in the next iteration/generation. This iteration is set 5 times which is equivalent to the repetition of the random node generating process in the path optimization space, so that reposition does not reduce the quota of the optimization itself. After the termination criteria are met, another beacon reposition is carried out. Gradually, the random appointment of beacons that are repositioned is what makes the proposal called the batch beacon optimization method. This process stores  $T_{opt}$  and as a note, the  $T_{opt}$  is  $T_{init}$  which has been optimized by applying Algorithm 7.  $T_{opt}$  consists of beacon nodes connecting  $x_{init}$  with  $x_{goal}$ . Because it implements a direct connection of nodes to the furthest nodes that is collision-free, the number of beacon nodes of this path is clearly less than all the nodes contained in  $T_{init}$  other than  $x_{init}$  and  $x_{goal}$ . The number of beacon nodes  $N$  is then considered as a decision variable in the optimization process using HBMO. The small number makes it relevant for HBMOs to carry out optimization by moving their locations to relevant areas. This displacement is limited by the radius  $r$  for all beacon nodes. Apart from that, the criteria for this decision variable are to adhere to a connection without collisions with obstacles in the environment. Furthermore, considering that the path from the beacon node represents a candidate solution, the fitness function used is the path cost which calculates the total connection distance from  $x_{start} - x_{goal}$  through the beacon nodes. Thus, for the purpose of generating optimal paths this type of optimization refers to minimizing the search space with limited time. Based on this time limitation, the HBMO generation is regulated according to the number of remaining samples towards the specified maximum sampling. HBMO with a deployment pattern like this allows the beacon node to shift to another area, with the potential to be connected directly to another farthest beacon node. So as to end the cycle of determining the best path, path optimization is carried out again at the end of the process before the final path is determined. Overall, this proposed method can be seen in the flowchart in Figure 2.

#### Algorithm 6. HBMO-optimized Fast-RRT\*

```

Input:  $x_{init}, X_{goal}, Map$ 
Output: A path  $T_{init}$  connecting  $x_{init}$  and  $x_{goal}$ 
for  $i = 1:N$ 
 $x_{rand} \leftarrow fastsampling(Map)$ 
 $x_{near} \leftarrow Near(T, x_{rand})$ 
 $x_{new} \leftarrow randomSteer(x_{near}, x_{rand})$ 
 $E_i \leftarrow Edge(x_{near}, x_{new})$ 
if  $CollisionFree(Map, E_i)$ 
 $Q \leftarrow Near(x_{new}, T, r)$ 
 $x_{min} \leftarrow Chooseparent(Q, x_{new}, x_{near}, E_i)$ 
 $T \leftarrow Connect(T, x_{min}, x_{new}, E_i)$ 
 $T \leftarrow Rewire(T, Q, x_{new})$ 
if  $x_{new} \in X_{goal}$ 
 $T_{init} \leftarrow getPath(T)$ 
 $T_{opt} \leftarrow optimizePath(T_{init}, Map)$ 
 $T_{opt} \leftarrow HBMO(T_{opt}, Map, i + 1)$ 
endif
endif
endfor

```

#### Algorithm 7. Optimize path

```

Input:  $Map, P$ 
Output:  $P_{opt}$ 
 $P = [ ]$ 
 $x_{check} = P_1$ 
for  $i = 1: length(P) - 1$ 
 $E_i \leftarrow Edge(x_{check}, P_i)$ 
if  $CollisionFree(Map, E_i)$ 
continue
else
 $subpath \leftarrow P_{1:k-1}$ 
 $x_{check} = P_{i-1}$ 
 $P_{opt} \leftarrow append(subpath)$ 
endif
endfor
 $P_{opt} \leftarrow append(P_{end})$ 

```

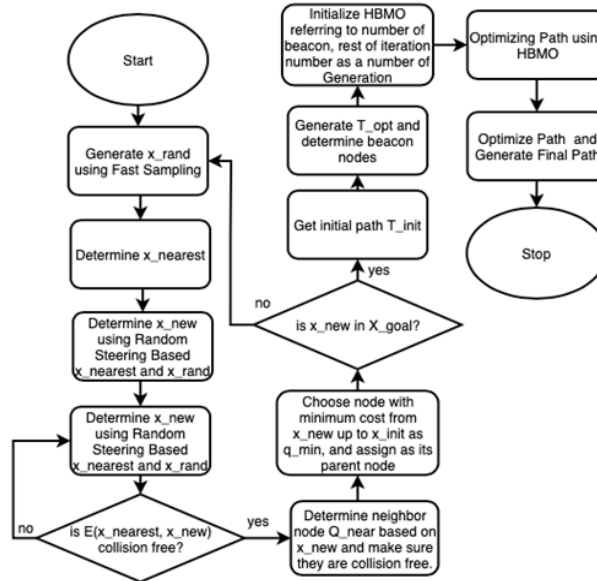


Figure 2. Flowchart of proposed algorithm

#### 4. RESULTS AND DISCUSSION

To verify effectiveness, the proposed method is applied to solve global path planning problems. There are 3 types of environmental models, categorized into maze environments with wide hallways and mazes with narrow hallways, which will be used for this trial. It is not only the success of the method that is to be tested through this experiment and testing, but also the validity of the effectiveness of the method in terms of optimality and convergence rate. This basis is the basis for testing which will involve comparisons with previous methods, such as RRT-Star, Fast-RRT, and HBMO-assisted Fast-RRT. To obtain a fair comparison and analysis, the parameterization and determination of start points, and goal points are equalized. The starting point is placed at (5, 75) and the goal point at (75, 5) with the number of samplings allowed set based on the complexity of the test environment. For a comparison of the performance of the three methods for the first environment, in Figure 3.

As shown in Figure 3, the three different methods can properly solve the problem with a sampling time of 1500. However, their sampling requirements to find the initial path are different to each other. The RRT\* requires 1203 repetitive sampling as can be seen in Figure 3(a), Fast-RRT requires 1094 as can be seen in Figure 3(b), and the proposed method is 1009 as can be seen in Figure 3(c). This shows that the proposed method is like Fast-RRT for the convergence rate value. This achievement is logical, because the proposed method applies the same techniques during exploration, namely, the application of fast sampling and random steering. However, referring to the cost path shown in Figure 3, the proposed method provides better values than the two predecessor methods. This proves that the proposed method can provide more optimal values even though there are sampling restrictions allowed during problem solving. As note, the red bullets are the beacon contained by path before the optimization conducted. The green bullets are the representative beacons after they are automatically adjusted by HBMO. According to the duration for all processes, the proposed method also faster than other involved methods in Table 1.

Up to this point, the HBMO shows its role to improve optimality of previous path. However, experiments on the 1st environment alone are not enough to prove that the proposed method is more effective. Therefore, a second experiment was carried out and the results can be seen visually in Figure 4. By continuing to use the permitted number of samples, namely 7000 times, the performance of the three methods was compared again. In this second environment, there are a number of passages with narrower distances and it is mandatory to pass them to be able to explore the area near the goal point. According to the required sampling time, RRT\* needs 6430 times of sampling to get the initial path, Fast-RRT needs 5463 times, and the proposed method 5346 times. The need for larger sampling shows that the RRT\* is having difficulty carrying out exploration by relying only on uniformly generated sampling as can be seen in Figure 4(a). So, it can be said that by maintaining the improved-RRT which includes Fast-Sampling and Random Steering, Fast-RRT and the proposed method can show better results. Furthermore, the path generated by RRT\* does not have sufficient optimality with quite high costs. Meanwhile, in Fast RRT, the optimality of the path is still not enough even though a large sampling has been completed. This is caused by the absence of a



rewiring process which can be useful for re-determining connections to nodes on the sub-path. Apart from that, the non-optimal path is also thought to be caused by the fusion step not being able to run properly due to the failure to get a new path that is better than the previous optimal path. Even though path optimization has been implemented, the final optimality is still lacking and worse compared to RRT\*, with path cost of 179.2243 as can be seen in Figure 4(b). It might be due to the influence of the previous optimality or no wiring process maintaining the quality of the previously obtained path. Meanwhile, for the proposed method in this second case, performance has shown effectiveness with a cost path of 147.4175 at the same number of samples as can be seen in Figure 4(c).

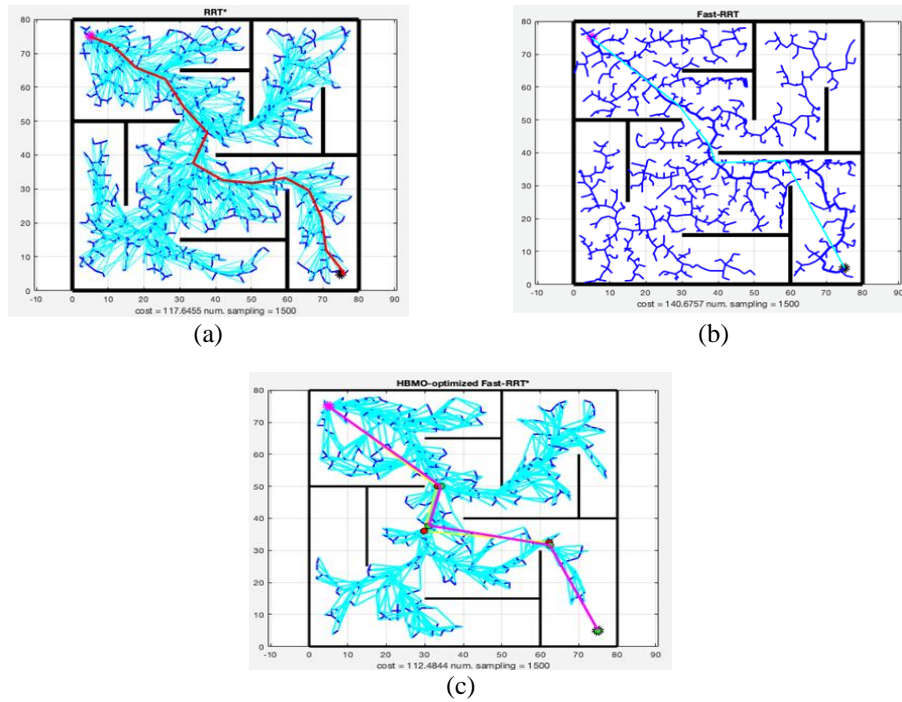


Figure 3. The performance of (a) RRT\*, (b) Fast-RRT, and (c) HBMO-optimized Fast-RRT

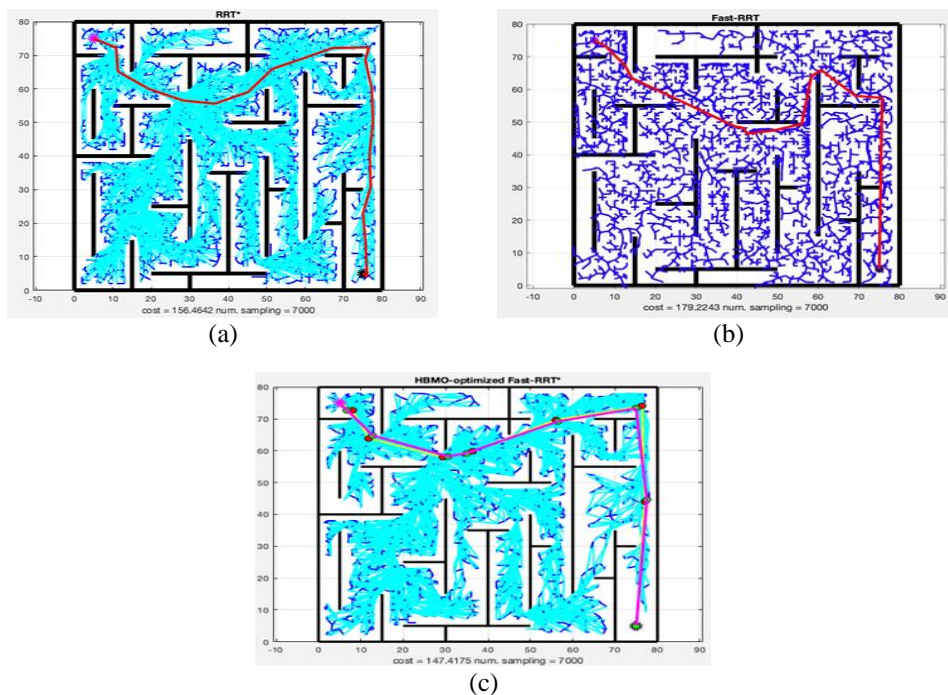


Figure 4. The performance of (a) RRT\*, (b) Fast-RRT, and (c) HBMO-optimized Fast-RRT

Table 1. Duration for solving the problem

Testing criteria	RRT*	Fast-RRT	HBMO-optimized Fast-RRT
Duration (second)	64.4352	59.1542	58.7744
Cost path	117.6455	140.6757	112.4844

Although by referring to the results in Figure 4 the effectiveness of the proposed method can already be proven, to increase confidence their performances are compared in term of duration need to solving path planning, as shown in Table 2. Moreover, to again validity the effectiveness of proposed method third experiment was conducted. Even though the starting and goal position are placed in the same position, the complexity of environment is increased. This complexity is represented by the presence of a greater number of narrow alleys compared to previous environments. Referring to this complexity, the number of samplings allowed is 7000 and the performance is examined in terms of optimality and convergence rate. The positioning of the starting and goal nodes in this third case refers to the need to test the exploration process through a narrow passage. The results of this experiment can be seen in Figure 5. In this experiment, RRT\* requires 6870 times of sampling to get the initial path (Figure 5(a)), Fast-RRT 6136 (Figure 5(b)), and the proposed algorithm requires 6090 times of sampling (Figure 5(c)).

Referring to Figure 5, the path cost given by the proposed method again shows the best value compared to RRT\* and Fast-RRT. Judging from the convergence rate, it has similarities with Fast-RRT which is clearly faster in the search process. So, it can be said that the proposed method can effectively adopt the advantages of Fast-RRT and the advantages of RRT\* in unified work. The precise use of RRT\* which is operated after fast-sampling and random sampling have been carried out sequentially, has a positive impact when environmental complexity is increased. Furthermore, by considering the optimality of RRT\* which provides more effective values compared to Fast-RRT, this precision is further justified. So, it is strong enough to say that the optimality of a path can be increased by including a rewiring process, which is not found in Fast-RRT. Next, to ensure that the working speed of the proposed method is also better than the others, the duration of solving the path planning problem is given in Table 3.

Table 2. Duration for solving the problem

Testing criteria	RRT*	Fast-RRT	HBMO-optimized Fast-RRT
Duration (second)	369.4352	253.9303	239.9303
Cost Path	156.4642	179.2243	147.4175

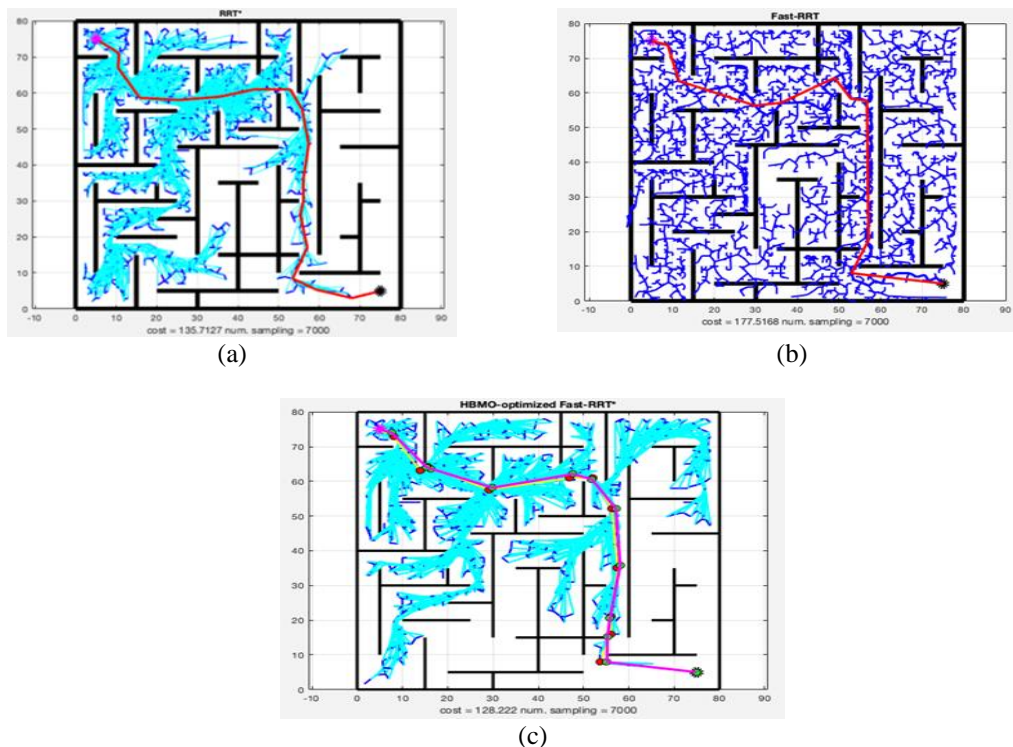


Figure 5. The performance of (a) RRT\*, (b) Fast-RRT, and (c) HBMO-optimized Fast-RRT

Table 3. Duration for solving the problem

Testing criteria	RRT*	Fast-RRT	HBMO-optimized Fast-RRT
Duration (second)	356.3721	297.7423	269.4368
Cost path	135.7127	177.5168	128.222

Referring to Table 3, although there is not a significant difference in the number of samplings required to solve the problem, Fast-RRT and the proposed method have succeeded in outperforming exploration speed. Based on Tables 1-3, it is found that the average value of the Fast-RRT cost path reduction is 21.85% at the level of environmental complexity tested. In addition, by looking at each decrease in the duration of the work, namely 0.64, 5.51, and 9.51 for each complexity, both from environment 1, environment 2, and environment 3, the average result of the decrease in completion time is 5.22%.

## 5. CONCLUSION

With fast-sampling, random steering and fast optimal, fast RRT offers effectiveness in terms of convergence rate and optimality in solving path planning problems. However, Path Fusion, which is the initial stage in fast-optimal, is very dependent on the uniqueness of the new path obtained so that fusion can then be carried out by involving the previous path. However, in an environment with complexity represented by the presence of narrow alleys, unique paths are rare and may not be available. This becomes even more of a challenge when the process of rewiring is not provided. So, the fast-optimal is not strong enough to produce an optimal path. Referring to these findings, RRT\* was inserted into Fast-RRT after the random steering process was carried out. This is intended to support obtaining sub-optimal paths before optimization is carried out. In contrast to Fast-RRT, the optimization offered in this paper is the application of HBMO with the objective of adjusting the beacon position to obtain lower path costs. And referring to the comparison results, the proposed method can provide better effectiveness values in terms of convergence rate and optimality.

## ACKNOWLEDGEMENTS




This research is supported by Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia through its competitive 2023's research funding, Schema of Regular Fundamental Research, in partnership with the Research Centre of Universitas Mercu Buana.

## REFERENCES




- [1] A. Adriansyah, H. Suwoyo, Y. Tian, and C. Deng, "Improving wall-following robot performance using PID-PSO controller," *Jurnal Teknologi*, vol. 81, no. 3, pp. 119–126, Apr. 2019, doi: 10.11113/jt.v81.13098.
- [2] Y. Tian, H. Suwoyo, W. Wang, and L. Li, "An ASVSF-SLAM algorithm with time-varying noise statistics based on MAP creation and weighted exponent," *Mathematical Problems in Engineering*, vol. 2019, no. 1, Jan. 2019, doi: 10.1155/2019/2765731.
- [3] H. Suwoyo, C. Deng, Y. Tian, and A. Adriansyah, "Improving a wall-following robot performance with a PID-genetic algorithm controller," in *International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, IEEE, Oct. 2018, pp. 314–318. doi: 10.1109/EECSI.2018.8752907.
- [4] H. Suwoyo *et al.*, "Maximum likelihood estimation-assisted ASVSF through state covariance-based 2D SLAM algorithm," *Telkommika (Telecommunication Computing Electronics and Control)*, vol. 19, no. 1, pp. 327–338, Feb. 2021, doi: 10.12928/TELKOMNIKA.V19I1.16223.
- [5] Y. Tian, H. Suwoyo, W. Wang, D. Mbemba, and L. Li, "An AEKF-SLAM algorithm with recursive noise statistic based on MLE and EM," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 97, no. 2, pp. 339–355, 2020, doi: 10.1007/s10846-019-01044-8.
- [6] S. G. Tzafestas, *Introduction to mobile robot control*. Elsevier, 2013. doi: 10.1016/C2013-0-01365-5.
- [7] M. R. H. Al-Dahhan and M. M. Ali, "Path tracking control of a mobile robot using fuzzy logic," in *13th International Multi-Conference on Systems, Signals and Devices, SSD 2016*, IEEE, Mar. 2016, pp. 82–88. doi: 10.1109/SSD.2016.7473656.
- [8] D. Fethi, A. Nemra, K. Louadj, and M. Hamerlain, "Simultaneous localization, mapping, and path planning for unmanned vehicle using optimal control," *Advances in Mechanical Engineering*, vol. 10, no. 1, Jan. 2018, doi: 10.1177/1687814017736653.
- [9] I. B. Jeong, S. J. Lee, and J. H. Kim, "Quick-RRT\*: triangular inequality-based implementation of RRT\* with improved initial solution and convergence rate," *Expert Systems with Applications*, vol. 123, pp. 82–90, Jun. 2019, doi: 10.1016/j.eswa.2019.01.032.
- [10] Q. Zhou and G. Liu, "UAV Path Planning Based on the Combination of A-star Algorithm and RRT-star Algorithm," *Proceedings of 2022 IEEE International Conference on Unmanned Systems, ICUS 2022*, pp. 146–151, 2022, doi: 10.1109/ICUS55513.2022.9986703.
- [11] M. A. Khan, "A comprehensive study of Dijkstra's algorithm," *SSRN Electronic Journal*, 2023, doi: 10.2139/ssrn.4559304.
- [12] M. A. Javaid, "Understanding Dijkstra algorithm," *SSRN Electronic Journal*, 2013, doi: 10.2139/ssrn.2340905.
- [13] I. Noreen, A. Khan, and Z. Habib, "A comparison of RRT, RRT\* and RRT\*-Smart path planning algorithms," *IJCSNS International Journal of Computer Science and Network Security*, vol. 16, no. 10, pp. 20–27, 2016, [Online]. Available: [http://cloud.politala.ac.id/politala/1.Jurusan/Teknik Informatika/19\\_e-journal/Jurnal Internasional TI/IJCSNS/2016 Vol. 16 No. 10/20161004\\_A Comparison of RRT, RRT\\* and RRT\\* - Smart Path Planning Algorithms.pdf](http://cloud.politala.ac.id/politala/1.Jurusan/Teknik Informatika/19_e-journal/Jurnal Internasional TI/IJCSNS/2016 Vol. 16 No. 10/20161004_A Comparison of RRT, RRT* and RRT* - Smart Path Planning Algorithms.pdf)

- [14] Z. Lee and X. Chen, "Path planning approach based on probabilistic roadmap for sensor based car-like robot in unknown environments," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, IEEE, 2004, pp. 2907–2912. doi: 10.1109/ICSMC.2004.1400774.
- [15] D. Zhang, Y. Xu, and X. Yao, "An improved path planning algorithm for unmanned aerial vehicle based on RRT-Connect," in *Chinese Control Conference, CCC*, IEEE, Jul. 2018, pp. 4854–4858. doi: 10.23919/ChiCC.2018.8483405.
- [16] J. Chen, Y. Zhao, and X. Xu, "Improved RRT-connect based path planning algorithm for mobile robots," *IEEE Access*, vol. 9, pp. 145988–145999, 2021. doi: 10.1109/ACCESS.2021.3123622.
- [17] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmady, and I. Ali, "Informed RRT\*-Connect: an asymptotically optimal single-query path planning method," *IEEE Access*, vol. 8, pp. 19842–19852, 2020. doi: 10.1109/ACCESS.2020.2969316.
- [18] K. Naderi, J. Rajamaki, and P. Hamalainen, "RT-RRT\*: a real-time path planning algorithm based on RRT\*," in *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games, MIG 2015*, New York, NY, USA: ACM, Nov. 2015, pp. 113–118. doi: 10.1145/2822013.2822036.
- [19] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Autonomous Robots*, vol. 40, no. 6, pp. 1079–1093, Aug. 2016. doi: 10.1007/s10514-015-9518-0.
- [20] D. Wu, L. Wei, G. Wang, L. Tian, and G. Dai, "APF-IRRT\*: an improved informed rapidly-exploring random trees-star algorithm by introducing artificial potential field method for mobile robot path planning," *Applied Sciences (Switzerland)*, vol. 12, no. 21, p. 10905, Oct. 2022. doi: 10.3390/app122110905.
- [21] X. Xinying, X. Jun, and X. Keming, "Path planning and obstacle-avoidance for soccer robot based on artificial potential field and genetic algorithm," in *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, IEEE, 2006, pp. 3494–3498. doi: 10.1109/WCICA.2006.1713018.
- [22] I. B. Jeong, S. J. Lee, and J. H. Kim, "RRT\*-Quick: a motion planning algorithm with faster convergence rate," in *Advances in Intelligent Systems and Computing*, vol. 345, 2015, pp. 67–76. doi: 10.1007/978-3-319-16841-8\_7.
- [23] Z. Yu and L. Xiang, "NPQ-RRT \*: an improved RRT \* approach to hybrid path planning," *Complexity*, vol. 2021, no. 1, Jan. 2021. doi: 10.1155/2021/6633878.
- [24] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT\*: an improved path planning algorithm for mobile robots," *Expert Systems with Applications*, vol. 152, p. 113425, Aug. 2020. doi: 10.1016/j.eswa.2020.113425.
- [25] F. S. Elkazzaz, M. A. H. Abozied, and C. Hu, "Hybrid RRT/DE algorithm for high performance UCAV path planning," in *ACM International Conference Proceeding Series*, New York, NY, USA: ACM, Dec. 2017, pp. 235–242. doi: 10.1145/3171592.3171618.
- [26] Q. Li, J. Wang, H. Li, B. Wang, and C. Feng, "Fast-RRT\*: an improved motion planner for mobile robot in two-dimensional space," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 17, no. 2, pp. 200–208, Feb. 2022. doi: 10.1002/tee.23502.
- [27] Z. Wu, Z. Meng, W. Zhao, and Z. Wu, "Fast-RRT: a RRT-based optimal path finding method," *Applied Sciences (Switzerland)*, vol. 11, no. 24, p. 11777, Dec. 2021. doi: 10.3390/app112411777.
- [28] S. Biswas and S. Acharyya, "Parameter estimation of gene regulatory network using honey bee mating optimization," in *Proceedings - 4th International Conference on Emerging Applications of Information Technology, EAIT 2014*, IEEE, Dec. 2014, pp. 3–8. doi: 10.1109/EAIT.2014.42.
- [29] O. Bozorg-Haddad, M. Solgi, and H. A. Loáiciga, *Meta-Heuristic and evolutionary algorithms for engineering optimization*. Wiley, 2017. doi: 10.1002/9781119387053.
- [30] M. Gavrilas, G. Gavrilas, and C. V. Sftines, "Application of honey bee mating optimization algorithm to load profile clustering," in *CIMSA 2010 - IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Proceedings*, IEEE, Sep. 2010, pp. 113–118. doi: 10.1109/CIMSA.2010.5611759.
- [31] K. L. Du and M. N. S. Swamy, *Search and optimization by metaheuristics*. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-41192-7.
- [32] Z. Zhou, B. Yuan, P. Xiao, and C. Zhang, "A Modified honey bees mating optimization algorithm for assembly line balancing problem," in *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*, IEEE, Dec. 2017, pp. 1–7. doi: 10.1109/SSCI.2016.7850274.
- [33] Q. Gu, X. Li, S. Jiang, and H. Mora, "Hybrid genetic grey wolf algorithm for large-scale global optimization," *Complexity*, vol. 2019, no. 1, Jan. 2019. doi: 10.1155/2019/2653512.
- [34] J. Liu, X. Wei, and H. Huang, "An improved grey wolf optimization algorithm and its application in path planning," *IEEE Access*, vol. 9, pp. 121944–121956, 2021. doi: 10.1109/ACCESS.2021.3108973.
- [35] N. Mittal, U. Singh, and B. S. Sohi, "Modified grey wolf optimizer for global engineering optimization," *Applied Computational Intelligence and Soft Computing*, vol. 2016, pp. 1–16, 2016. doi: 10.1155/2016/7950348.
- [36] J. Zhao and Z. M. Gao, "An improved grey wolf optimization algorithm with multiple tunnels for updating," *Journal of Physics: Conference Series*, vol. 1678, no. 1, p. 012096, Nov. 2020. doi: 10.1088/1742-6596/1678/1/012096.
- [37] H. Suwoyo, M. H. I. Hajar, P. Indriyanti, and A. Febriandirza, "The use of fuzzy logic controller and artificial bee colony for optimizing adaptive SVSF in robot localization algorithm," *Sinergi (Indonesia)*, vol. 28, no. 2, pp. 231–240, Apr. 2024. doi: 10.22441/sinergi.2024.2.003.
- [38] C. Chen, H. Du, and S. Lin, "Mobile robot wall-following control by improved artificial bee colony algorithm to design a compensatory fuzzy logic controller," in *ECTI-CON 2017 - 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, IEEE, Jun. 2017, pp. 856–859. doi: 10.1109/ECTICon.2017.8096373.
- [39] J. A. Abdor-Sierra, E. A. Merchán-Cruz, F. A. Sánchez-Garfias, R. G. Rodríguez-Cañizo, E. A. Portilla-Flores, and V. Vázquez-Castillo, "Particle swarm optimization for inverse kinematics solution and trajectory planning of 7-dof and 8-dof robot manipulators based on unit quaternion representation," *Journal of Applied Engineering Science*, vol. 19, no. 3, pp. 592–599, 2021. doi: 10.5937/jaes0-30557.
- [40] S. Agrawal and V. Shrivastava, "Particle swarm optimization of BLDC motor with fuzzy logic controller for speed improvement," in *8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017*, IEEE, Jul. 2017, pp. 1–5. doi: 10.1109/ICCCNT.2017.8204006.
- [41] N. Rokbani and A. M. Alimi, "Inverse kinematics using particle swarm optimization, a statistical analysis," *Procedia Engineering*, vol. 64, pp. 1602–1611, 2013. doi: 10.1016/j.proeng.2013.09.242.
- [42] S. Dereli and R. Köker, "IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator," *Sigma Journal of Engineering and Natural Sciences*, vol. 36, no. 1, pp. 77–85, 2018.




**BIOGRAPHIES OF AUTHORS**

**Heru Suwoyo**    is a lecturer at Department of Electrical Engineering, Universitas Mercu Buana, Indonesia. He received his Ph.D. from Department of Mechatronic Engineering, School of Mechatronic Engineering and Automation, Shanghai University, China. His research interest includes behavior based robotic, path planning and path tracking, mobile robot navigation, simultaneous localization and mapping, fuzzy logic controller, probability-based filtering method such as extended Kalman filter and smooth variable structure filter, adaptive-filtering and population-based metaheuristic optimization such as ant colony optimization, genetic algorithm, and particle swarm optimization. He can be contacted at email: heru.suwoyo@mercubuana.ac.id.






**Yingzhong Tian**    is a professor at School of Automation and Mechatronic Engineering, Shanghai University. He is also a Director of Research in LIMAR (Lab of Intelligent Mechanism and Advanced Robot) research team at Shanghai University. He obtained a Ph.D. degree in Mechanical manufacture and automation in 2007 from Shanghai University. His research interests include mobile robot, bionic robot, multi-robot systems, and image processing. He can be contacted at email: troytian@shu.edu.cn.



**Andi Adriansyah**    is a professor at Department of Electrical Engineering, Universitas Mercu Buana, Jakarta. His research interest includes automatic system, autonomic system, internet of things, mobile robot, robot movement, solar power, access rights, acquisition cycle, activation of system, active control, active switches, adaptive neuro-fuzzy inference system, amount of fertilizer, android application, angle difference, angular velocity, architectural forms, artificial neural network, automated guided vehicles, automatic operation, automatic processing, autonomic control, average response time, bill payments, and boost converter. He can be contacted at email: andi@mercubuana.ac.id.



**Julpri Andika**    is a lecturer at Department of Electrical Engineering, Universitas Mercu Buana, Indonesia. He received his master degree from Department of Mechanical Engineering, Beijing Institute of Technology, China in 2016. Besides that, he is currently Head of Research and Community Services Department. His research interest includes arm robotic, path planning, localization using particle filter, and cubature Kalman filter, and mobile robot and internet of things, closed-loop controller used for behavior-based robotic such as pid controller, fuzzy logic controller, h-infinity, and model predictive control. He can be contacted at email: julpri.andika@mercubuana.ac.id.