Muhammad Rajasa

Path Planning Algorithm Using the Hybridization of the Rapidly-Exploring Random Tree and Ant Colony Systems

Draft IEEE

Draft Jurnal IEEE

Universitas Komputer Indonesia

Document Details

Submission ID
trn:oid:::1:3636371645

Submission Date
Aug 30, 2026, 9:34 PM GMT+7

Download Date
Aug 30, 2026, 9:40 PM GMT+7

File Name
the_Rapidly-Exploring_Random_Tree_and_Ant_Colony_Systems_1.pdf

File Size
2.7 MB

17 Pages

11,060 Words

57,096 Characters

22% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Exclusions

- 100 Excluded Sources

Match Groups

160 Not Cited or Quoted 20%
Matches with neither in-text citation nor quotation marks

18 Missing Quotations 2%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

16% Internet sources
17% Publications
6% Submitted works (Student Papers)

Integrity Flags

1 Integrity Flag for Review

Replaced Characters
22 suspect characters on 2 pages
Letters are swapped with similar characters from another alphabet.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **160** Not Cited or Quoted 20%
  Matches with neither in-text citation nor quotation marks
- **18** Missing Quotations 2%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

16% Internet sources
17% Publications
6% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
Olympic College		<1%
2	Internet	
www-emerald-com-443.webvpn.sxu.edu.cn		<1%
3	Student papers	
iGroup		<1%
4	Internet	
doczz.net		<1%
5	Publication	
H.B. Gooi, Y.Q. Wang. "Efficient ordering algorithms for sparse matrix/vector met...		<1%
6	Publication	
Wenjuan Wang, Jiaye Li, Zongning Bai, Zhonghua Wei, Jingxuan Peng. " Toward O...		<1%
7	Internet	
michalcap.net		<1%
8	Internet	
www.aimspress.com		<1%
9	Internet	
wiredspace.wits.ac.za		<1%
10	Internet	
journal2.uad.ac.id		<1%

11	Internet	
www.ijeei.org		<1%
12	Internet	
backend.orbit.dtu.dk		<1%
13	Internet	
ebin.pub		<1%
14	Publication	
Devin Connell, Hung Manh La. "Dynamic path planning and replanning for mobil...		<1%
15	Internet	
research.ijcaonline.org		<1%
16	Publication	
Felix Burget, Maren Bennewitz, Wolfram Burgard. " BI RRT*: An efficient samplin...		<1%
17	Student papers	
Liverpool John Moores University		<1%
18	Publication	
Yaqub A. Prabowo, Bambang R. Trilaksono, Egi M. I. Hidayat, Brian Yuliarto. "Inte...		<1%
19	Internet	
dspace.thapar.edu:8080		<1%
20	Publication	
Reza Mashayekhi, Mohd Yamani Idna Idris, Mohammad Hossein Anisi, Ismail Ah...		<1%
21	Internet	
backoffice.biblio.ugent.be		<1%
22	Internet	
webpages.iust.ac.ir		<1%
23	Internet	
www.kostasalexis.com		<1%
24	Publication	
Advances in Intelligent Systems and Computing, 2014.		<1%

25 Publication

Haotian Li, Yiting Kang, Haisong Han. "Dynamic Informed Bias RRT*-Connect: Im... <1%

26 Student papers

The Robert Gordon University <1%

27 Internet

elvedit.com <1%

28 Publication

Miao Zhang, Yiming Wang, Junsheng Gao, Ziju Li, Xiaoguang Han. "An improved R... <1%

29 Internet

www.repository.cam.ac.uk <1%

30 Internet

mavrovouniotis.github.io <1%

31 Internet

iranarze.ir <1%

32 Internet

psecommunity.org <1%

33 Publication

Rahmi Amtha, Ferry Sandra, Rosalina Tjandrawinata, Indrayadi Gunardi, Anggrae... <1%

34 Internet

s3-ap-southeast-1.amazonaws.com <1%

35 Publication

K. Damodaran. "A high resolution, integrated data acquisition system for aerospa... <1%

36 Internet

espace.curtin.edu.au <1%

37 Internet

pdffox.com <1%

38 Internet

sersc.org <1%

39 Internet
asrl.utias.utoronto.ca <1%

40 Internet
d-nb.info <1%

41 Internet
eprints.kingston.ac.uk <1%

42 Internet
opus.lib.uts.edu.au <1%

43 Internet
scholarworks.unist.ac.kr <1%

44 Internet
sro.sussex.ac.uk <1%

45 Internet
vtechworks.lib.vt.edu <1%

46 Internet
www.qichegongcheng.com <1%

47 Publication
Ari Wibowo, Bambang Riyanto Trilaksono, Egi Muhammad Idris Hidayat, Rinaldi ... <1%

48 Publication
Kuo, Yu-Cheng. "Insertion ant colony optimization for solving the traveling sales... <1%

49 Internet
ejournal.kjpupi.id <1%

50 Internet
repository.tudelft.nl <1%

51 Internet
scholar.sun.ac.za <1%

52 Internet
vdoc.pub <1%

53 Publication
Engineering Computations, Volume 25, Issue 2 (2008-03-09) <1%

54 Publication
Zengxin Chen, Zhihao Luo, Xiaojie Jin, Jianmai Shi. "Multi-UAV path planning probl... <1%

55 Publication
A. Broggi, M. Cellario, P. Lombardi, M. Porta. "An evolutionary approach to visual ... <1%

56 Internet
ethesis.inp-toulouse.fr <1%

57 Internet
ric.zp.edu.ua <1%

58 Student papers
CSU, Long Beach <1%

59 Student papers
Jamia Milia Islamia University <1%

60 Publication
Marco Dorigo. "<![CDATA[Ant colony optimization]]>", IEEE Computational Intellig... <1%

61 Student papers
The University of Manchester <1%

62 Internet
iridia0.ulb.ac.be <1%

63 Student papers
National University of Singapore <1%

64 Internet
hub.hku.hk <1%

65 Publication
R. Montemanni, D.H. Smith, S.M. Allen. "An ANTS algorithm for the minimum-spa... <1%

66 Publication
Wu Deng, Huimin Zhao, Li Zou, Guangyu Li, Xinhua Yang, Daqing Wu. "A novel col... <1%

67 Internet
cs-courses.mines.edu <1%

68 Internet
soil.copernicus.org <1%

69 Publication
Devaurs, Didier, Thierry Simeon, and Juan Cortes. "Optimal Path Planning in Com... <1%

70 Publication
Islam, Fahad, Jauwairia Nasir, Usman Malik, Yasar Ayaz, and Osman Hasan. "RRT<... <1%

71 Internet
gaoxifeng.github.io <1%

72 Internet
mdpi.com <1%

73 Internet
nzdr.ru <1%

74 Internet
polibits.ojs.gelbukh.com <1%

75 Internet
www.dsg.cs.tcd.ie <1%

76 Internet
www.semanticscholar.org <1%

77 Publication
"Intelligent Computing Theories and Application", Springer Science and Business ... <1%

78 Publication
Bud Fox, Wei Xiang, Heow Pueh Lee. "Industrial applications of the ant colony opt... <1%

79 Publication
Chyon Hae Kim, Hiroshi Tsujino, Shigeki Sugano. "Rapid approximation for optim... <1%

80 Publication
Mingyue Zhang, Yifan Chen, Sha lUO, Qingdang Li, Hui Zhao, Linan Zu. "Path Plan... <1%

81	Publication	P. Pharpatara, B. Herisse, Y. Bestaoui. "3-D Trajectory Planning of Aerial Vehicles ... <1%
82	Internet	app.ait.kyushu-u.ac.jp <1%
83	Internet	code.ulb.ac.be <1%
84	Internet	ijeecs.iaescore.com <1%
85	Internet	jestec.taylors.edu.my <1%
86	Internet	reference-global.com <1%
87	Internet	repozitorij.unizg.hr <1%
88	Internet	riunet.upv.es <1%
89	Internet	semantjournals.org <1%
90	Internet	staff.washington.edu <1%
91	Internet	www.cambridge.org <1%
92	Internet	www.mkhasan.info <1%
93	Publication	Chuanrong Huang, Bingjie Tang, Zhiyang Guo, Qi Su, Jingyao Gai. "Agile-RRT*: A F... <1%
94	Publication	Devin Connell, Hung Manh La. "Extended rapidly exploring random tree–based d... <1%

95	Publication	
Gang Wang. "<![CDATA[Ant Colony Optimizations for Resource- and Timing-Const...		<1%
96	Publication	
Hasan, Yazied A.. "Mapping Homogeneous Configuration States for Learning Bas...		<1%
97	Publication	
J. M. Cecilia, J. M. Garcia, M. Ujaldon, A. Nisbet, M. Amos. "Parallelization strategie...		<1%
98	Publication	
S.L. Ho. "A Modified Ant Colony Optimization Algorithm Modeled on Tabu-Search ...		<1%
99	Publication	
Wenli Lei, Yang Lei, Hongbo Guo, Kun Jia. "Research on energy-saving relay select...		<1%
100	Internet	
academic.oup.com		<1%
101	Internet	
cronfa.swan.ac.uk		<1%
102	Internet	
cyberleninka.org		<1%
103	Internet	
deepai.org		<1%
104	Internet	
dlibrary.univ-boumerdes.dz:8080		<1%
105	Internet	
ejbio.imedpub.com		<1%
106	Internet	
etheses.bham.ac.uk		<1%
107	Internet	
eventkampus.com		<1%
108	Internet	
harmony-eu.org		<1%

109 Internet
rcastoragev2.blob.core.windows.net <1%

110 Internet
www.freepatentsonline.com <1%

111 Internet
www.osti.gov <1%

112 Internet
www.researchsquare.com <1%

113 Publication
"Swarm Intelligence", Springer Science and Business Media LLC, 2010 <1%

114 Publication
Dan Feng. "Ant System for a Multi-vehicle Routing Problem", 2007 International C... <1%

115 Publication
Liang Ma, Jianru Xue, Kuniaki Kawabata, Jihua Zhu, Chao Ma, Nanning Zheng. "Eff... <1%

116 Publication
Lecture Notes in Computer Science, 2002. <1%

117 Publication
Mohammad Rzea Jabbarpour, Houman Zarrabi, Jason J. Jung, Pankoo Kim. "A Gre... <1%

118 Publication
Qureshi, Ahmed Hussain, Saba Mumtaz, Yasar Ayaz, Osman Hasan, Mannan Saee... <1%

119 Publication
Rina Ristiana, Arief Syaichu Rohman, Carmadi Machbub, Agus Purwadi, Estiko Rij... <1%

120 Publication
Wenzheng Chi, Chaoqun Wang, Jiankun Wang, Max Q.-H. Meng. "Risk-DTRRT-Base... <1%

121 Publication
Z. Sun. "Narrow Passage Sampling for Probabilistic Roadmap Planning", IEEE Tran... <1%

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 12, 2021, accepted November 4, 2021, date of publication November 11, 2021,
date of current version November 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3127635

Path Planning Algorithm Using the Hybridization of the Rapidly-Exploring Random Tree and Ant Colony Systems

MUHAMMAD ARIA RAJASA POHAN[1], BAMBANG RIYANTO TRILAKSONO[1,2], (Member, IEEE), SIGIT PUJI SANTOSA[3], AND ARIEF SYAICHU ROHMAN[1], (Member, IEEE)

[1]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung 40132, Indonesia
[2]University Center of Excellence—Artificial Intelligence on Vision, NLP and Big Data Analytics (U-CoE AI-VLB), Institut Teknologi Bandung, Bandung 40132, Indonesia
[3]Faculty of Mechanical and Aerospace Engineering, Institut Teknologi Bandung, Bandung 40132, Indonesia

Corresponding author: Bambang Riyanto Trilaksono (briyanto@lskk.ee.itb.ac.id)

This work was supported in part by the Indonesian Ministry of Finance under the RISPRO LPDP (Pendanaan Riset Inovatif Produktif, Lembaga Pengelola Dana Pendidikan) Research Funding, Institut Teknologi Bandung under P2MI (Penelitian Pengabdian Masyarakat dan Inovasi) Research Program, and the Indonesian Ministry of Education, Culture, Research and Technology under the World Class University (WCU) Program managed by Institut Teknologi Bandung and under the World Class Research (WCR) Program.

ABSTRACT This paper proposes a path planning algorithm using the hybridization of the rapidly-exploring random tree (RRT) and ant colony system (ACS) algorithms. The RRT algorithm can quickly generate paths. However, the resulting path is suboptimal. Meanwhile, the ACS algorithm can generate the optimal path from the suboptimal previous path information. Then, the proposed algorithm will combine the advantages of RRT with the ACS algorithm. Therefore, it can reach the optimal value with a good convergence speed. We call this proposed algorithm the RRT-ACS algorithm. This study developed a new method for hybridizing the RRT and ACS algorithms for path planning problems. This hybridization process is carried out using one of the ACS principles: the pseudorandom proportional rule. The performance of the proposed algorithm with the RRT*, informed RRT*, RRT*-connect, and informed RRT*-connect algorithms is tested with several benchmark cases. The test results from benchmark case tests with known optimal values indicate that the proposed algorithm has succeeded in achieving those optimal values. Furthermore, statistical tests have also been carried out to verify whether there is a significant difference in performance between the RRT-ACS algorithm and the existing algorithms. The test and statistical analysis results show that the RRT-ACS algorithm has good performance and convergence speed. We also discuss the stability, robustness, convergence, and rapidity of the RRT-ACS algorithm. The results indicates that the RRT-ACS algorithm may be used in applications that require fast and optimal path planning algorithms such as robots and autonomous vehicles.

INDEX TERMS Path planning, rapidly-exploring random tree, ant colony system, convergence speed.

I. INTRODUCTION

The path planning problem is defined as finding a path in the configuration space while satisfying a set of constraints [1]. Path planning has applications in various fields, such as self-driving cars, medical applications, graphical animation, cell transportation, and robotic surgery [2]–[6]. Path planning algorithms can be divided into several main classes: variational methods, graph search methods, and incremental search methods [1], [7]. Graph search methods such as Djikstra and A* are often resolution-complete and resolution optimal [8]. This means that the completeness and optimality of the algorithm can only be guaranteed if the resolution of the discretization of the state space is good enough. However, good resolution may be challenging to achieve in high-dimensional spaces [9]. The incremental search method uses a sample-based method to avoid the need for the discretization of state space. This makes the incremental search method

The associate editor coordinating the review of this manuscript and approving it for publication was Qichun Zhang.

more effective with the size of the problem [10]. Examples of incremental search methods are rapidly-exploring random tree (RRT) [11] and RRT* [12]. The advantage of the RRT algorithm is that it can provide rapid solutions in multidimensional systems [13]. However, it has a disadvantage since it only offers a suboptimal solution [14]. Karaman and Frazzoli then developed the RRT algorithm into RRT* [15], which provides an asymptotically optimal solution [16]. Karaman [14] reports that the RRT* algorithm can converge to the optimal solution when the RRT algorithm only converges to a $\sqrt{2}$ factor of the optimal solution. However, it has the disadvantage of a slow convergence rate [17]–[19]. This paper focuses on developing a path planning algorithm based on the RRT algorithm to achieve optimal values with good convergence speeds.

Many methods have been developed to improve the quality of the RRT and RRT* algorithms. Kuffner and Lavalle introduce a dual-tree version of RRT, namely RRT-connect [20]. RRT-connect incrementally grows two trees simultaneously, one from the starting node and the other from the goal node. These two trees aggressively try to find a connection. RRT-connect could find solutions faster than RRT, especially in narrow passages that the planner has to cross to find solutions. Kang *et al.* [21] reports that the RRT-connect algorithm has an average performance ratio that is 62% better than that of the RRT algorithm. However, RRT-connect cannot provide the optimal solution. Klemm *et al.* introduced the RRT*-connect algorithm [22], which combines RRT-connect with RRT*. RRT*-connect is a dual-tree version of RRT*, providing the asymptotic optimal solution. Although RRT*-connect can provide the asymptotic optimal solution, it must search all states to optimize their paths. In [10], Gammel *et al.* introduced the informed RRT* algorithm, which uses informed sampling on RRT* after a first solution is found. The sampling process is performed in the ellipse area surrounding the starting node with the goal node. Wang *et al.* [23] reports that the informed RRT* algorithm can achieve the optimal solution 1.2–10.3× faster than the RRT* algorithm in the case of a simple environment, gap environment, random environment, and half-enclosed environment. However, informed RRT* has a problem when the goal node is hidden behind a narrow passage. In this situation, informed RRT* will take a longer time to reach the goal node. Mashayekhi *et al.* [24] introduced the informed RRT*-connect algorithm. Informed RRT*-connect behaves like RRT*-connect until a first path is found. Once a first solution is found, the sampling area of informed RRT*-connect is limited to an ellipsoid subset of the state space whose eccentricity depends on the length of the shortest current solution. Research to improve the quality of the RRT and RRT* algorithms is still being conducted to the authors' knowledge.

The ant colony system (ACS) algorithm was proposed by Dorigo and Gambardella [25], [26]. The ACS algorithm is an extension of the ant colony optimization (ACO) algorithm [27]. The ACO algorithm can generate the optimal path from the previous suboptimal path information [28], [29]. Several previous studies have discussed the implementation of the ACO algorithm in path planning [30]–[34]. However, these implementations require a discretization of the state space. There is a weakness when using discretization in the search environment, which will reduce performance in the higher dimensions [35]. Viseras *et al.* [36] proposed a path planning algorithm with ants for continuous domains using RRT* and $ACO_{\mathbb{R}}$. $ACO_{\mathbb{R}}$ is an extension of the ACO algorithm for continuous domains [37]. To the best of the authors' knowledge, no previous publications discuss using the original ACS with the RRT algorithm for path planning problems.

The contribution of this paper over existing relevant works in this area, in particular [10], [14], [22], [24], [36], is that it proposes a new method for hybridizing the RRT and ACS algorithms for path planning problems. We call this proposed algorithm the RRT-ACS algorithm. The hybridization process of the RRT and ACS algorithms is carried out using one of the ACS principles, the pseudorandom proportional rule. A random variable value will be generated at each iteration to determine whether the exploration or exploitation process will be carried out. If the exploitation process is selected, the determination of the new node will be based on the pheromone value. However, if the exploration process is chosen, the determination of the new node will be based on the RRT algorithm.

The novelty of this proposed algorithm is that it improves the performance of the RRT algorithm to reach the optimal value with a good convergence speed. This convergence speed can be increased by limiting the search area through the pheromone distribution. The motivation for using pheromone information in determining the new node is based on the principle of learning from experience [36], [38]. After each sampling, the contribution of a sample will be evaluated to improve the quality of the resulting path. These evaluation results will affect the sampling process in the next iteration.

Then, the performance of the proposed algorithm is compared against the RRT*, informed RRT*, RRT*-connect, and informed RRT*-connect algorithms using several benchmark cases. The benchmark cases used are clutter, tough passages, square field, multiple narrow passages, maze and trap. A comparative test between RRT-ACS and other existing algorithms indicates that the RRT-ACS algorithm has a good convergence speed. The proposed algorithm reached those optimal value based on the test results from some benchmark cases with known optimal values. Statistical analysis was also performed to verify a significant difference in performance between the RRT-ACS algorithm and other existing algorithms. The statistical analysis results show that the RRT-ACS algorithm has a good convergence speed.

The remainder of this paper is organized as follows. Section II describes some preliminaries. Section III presents the proposed algorithm: the RRT-ACS algorithm. Section IV provides an evaluation and discussion of the performance of the RRT-ACS algorithm compared to several variants of the RRT* algorithm. Statistical analysis of the simulation results

was also carried out. This section also discusses the stability, robustness, convergence and rapidity of the RRT-ACS algorithm. Finally, Section V presents some conclusions.

II. PRELIMINARIES

A. ANT COLONY SYSTEM ALGORITHM

The ACO algorithm (first introduced by Dorigo and Di Caro [39]) is a technique to resolve complicated combinatorial optimized problems. The ACO algorithm is based on the foraging behavior of an ant seeking a path between its colony and a source of food.

ACS was introduced by Dorigo and Gambardella [25] to improve the performance of the ACO. Jangra and Kait [40] showed that the ACS algorithm is better than the ACO in terms of the cost of the route generated and the computational time required. Zhang *et al.* [30] reported that the ACS algorithm could produce paths with an average length of 37.4% better than the ACO algorithm. Zhang also showed that the ACS algorithm could produce an average path length of 6.4% - 9.8% better than the average path length produced by two variations of the ACO algorithm, namely RAS and PS-ACO. Even when compared to other optimization algorithms, the ACS algorithm still shows its superiority. Dorigo and Gambardella [25] reported that the ACS algorithm could produce a 2.2% better path length than the genetic algorithm (GA), 1.3% better than evolutionary programming (EP), and 7.8% better than simulated annealing (SA). Because the ACS algorithm is better than the ACO in terms of the cost of the route generated and the computational time required, we use the ACS algorithm for hybridization with the RRT algorithm.

The main idea of the ACS algorithm is to make a group of ants work together. Each ant searches for a good solution in parallel and cooperates through indirect communication via pheromones. The ACS works as follows:

1) m ants are initially positioned on n nodes chosen according to some initialization rule
2) Each ant builds a tour
3) While constructing its tour, an ant also modifies the amount of pheromone on the visited edges by applying the local updating rule
4) Once all ants have terminated their tour, the amount of pheromone on edges is modified by applying the global updating rule

By building their tours, ants are guided by both heuristic information and pheromone information. An edge with a high amount of pheromones is a very desirable choice.

There are three main differences between ACS and ACO. First, the ACS algorithm exploits the search experience accumulated by ants more strongly than the ACO. The decision rule for ants k to move from node i to node j is determined by the pseudorandom proportional rule. The pseudorandom proportional rule equation is shown in Equation (1) [24].

$$j = \begin{cases} argmax_{l \epsilon \mathcal{N}_i^k} \left\{ \tau_{il} \left[\eta_{il} \right]^{\beta} \right\}, & if\ q \leq q_0; \\ J, & otherwise \end{cases} \tag{1}$$

where q is a uniformly distributed random variable in [0, 1], q_o $(0 \leq q_o \leq 1)$ is a parameter, and J is a random variable chosen according to the probability distribution given by Equation (2).

$$p_{ij}^k = \frac{\left[\tau_{ij} \right]^{\alpha} \left[\eta_{ij} \right]^{\beta}}{\sum_{l \epsilon \mathcal{N}_i^k} \left[\tau_{il} \right]^{\alpha} \left[\eta_{il} \right]^{\beta}}, \quad if\ j \in \mathcal{N}_i^k \tag{2}$$

where p_{ij}^k is the probability that ant k in node i chooses to go to node j, τ_{ij} is a pheromone trail, $\eta_{ij} = 1/d_{ij}$ is a heuristic value available a priori, α and β are two parameters that determine the relative influence of the pheromone trail and the heuristic information, and N_i^k is the feasible neighborhood of ant k when at node i.

The parameter q_o determines the relative importance of the exploitation compared to the exploration of the search space. With probability q_o, the ant moves to node j for which the product between the pheromone trail and heuristic information is maximized (exploitation process). With probability $1 - q_o$, the ant performs a biased exploration in which the probability is the same as in the ant system algorithm (as shown in Equation 2).

Another difference between the ACS and ACO is that pheromone evaporation and deposition only occur on the arcs belonging to the best-so-far tour. Thus, the pheromone update in the ACS is implemented by Equation 3.

$$\tau_{ij} \leftarrow (1 - \rho)\, \tau_{ij} + \rho {\tau_{ij}}^{bs}, \quad \forall\, (i, j)\, \epsilon T^{bs} \tag{3}$$

where $0 \leq \rho \leq 1$ is the pheromone evaporation rate, and $\Delta \tau_{ij}^{bs} = 1/C^{bs}$. Since evaporation and new pheromone deposits only apply to the arcs belonging to the best-so-far tour, the computational complexity of updating pheromones at each iteration is reduced.

The third difference between the ACS and ACO is the use of a local pheromone update during tour construction. Whenever an ant uses an arc (i, j) to move from node i to node j, it removes some pheromone from the arc to increase the path exploration alternative. The local pheromone update is implemented by Equation 4.

$$\tau_{ij} \leftarrow (1 - \xi)\, \tau_{ij} + \xi \tau_0 \tag{4}$$

where ξ $(0 < \xi < 1)$ and τ_o are two parameters. The effect of the local update rule is that every time an ant uses an arc (i, j), its pheromone trail τ_{ij} is reduced. Therefore, the arc becomes less desirable for another ant.

B. RRT AND RRT** ALGORITHMS

RRT is a sampling-based path planning algorithm usually applied to single-query path planning problems. RRT algorithms are designed to be used in continuous path planning problems and can be applied to high-dimensional space cases. The RRT algorithm incrementally extends a tree from the start configuration over the collision-free part of the state space and stops exploring once the tree reaches the goal configuration. The standard version of the RRT algorithm can be explained as follows.

1) Initialization of the tree with q_{init}
2) Iteratively, a random configuration (q_{rand}) is generated using a uniform sampling method in the configuration space. The new configuration (q_{rand}) needs to be checked to determine whether it is in free space.
3) If the new configuration is in free space, then the vertex of the tree that is closest to the new configuration (q_{near}) is searched.
4) One q_{near} has been found, and then the tree is extended toward q_{rand} from q_{near}. It will return the new vertex q_{new}.
5) Then, the motion toward q_{new} is checked to determine whether it is collision-free. If no collision occurs, then q_{new} can be saved in the tree.
6) Then, q_{new} is checked to determine whether it is in the goal region, and if so, the algorithm is terminated. If the goal region has not been reached, then the iteration is repeated.

The RRT algorithm has several characteristics: rapid exploration [41], monotone convergence, and probabilistic completeness [42]. However, the RRT algorithm is not asymptotic optimal. Therefore, Karaman developed the RRT algorithm into the RRT*, which has asymptotic optimal properties [14].

The basic algorithm of RRT* is shown in Algorithm 1. The main differences between the RRT and RRT* algorithms are the *chooseparent* (Lines 11 in Algorithm 1) and *rewiring tree* (Lines 13 in Algorithm 1) operations. The *chooseparent* algorithm is shown in Algorithm 2, and the *rewiring tree* algorithm is shown in Algorithm 3. The *chooseparent* algorithm will select the node around q_{new} that has the shortest distance to q_{start} (referred to as q_{min} in Line 7 in Algorithm 1). This q_{min} node will be the parent of q_{new}. Therefore, q_{new} will always have the optimal distance to q_{start}.

Algorithm 1 $X^{bs} \leftarrow \text{RRT}\left(q_{init}, q_{goal}, map\right)$

1: % ========== **Initialization**
2: $T \leftarrow InitializeTree()$
3: $T \leftarrow InsertNode\left(\emptyset, q_{init}, T\right)$
4: **for** $k = 1$ to N **do**
5: $q_{rand} \leftarrow RandomSample\left(k\right)$
6: $Q_{near} \leftarrow Near\left(T, q_{rand}\right)$
7: $q_{nearest} \leftarrow NearestNeighbor\left(q_{rand}, Q_{near}, T\right)$
8: $q_{new} \leftarrow Steer\left(q_{nearest}, q_{rand}, \Delta q\right)$
9: **if** $Obstaclefree\left(q_{nearest}, q_{new}\right)$ **then**
10: $Q_{near} \leftarrow Near\left(T_A, q_{new}\right)$
11: $q_{min} \leftarrow ChooseParent\left(q_{new}, Q_{near}, q_{nearest}\right)$
12: $T \leftarrow InsertNode\left(q_{min}, q_{new}, T\right)$
13: $T \leftarrow Rewire\left(T, Q_{near}, q_{min}\right)$
14: **end if**
15: **if** $CanConnected\left(q_{goal}, T\right)$ **then**
16: $X^{bs} \leftarrow UpdateBestPath\left(T, q_{goal}\right)$
17: **end if**
18: **end**

Algorithm 2 $q_{min} \leftarrow ChooseParent\left(q_{rand}, Q_{near}, q_{nearest}\right)$

1: $q_{min} \leftarrow q_{nearest}$
2: $c_{min} \leftarrow Cost\left(q_{nearest}\right) + c\left(q_{rand}\right)$
3: **for** $q_{near} \in Q_{near}$ **do**
4: $q_{path} \leftarrow Steer\left(q_{near}, q_{rand}, \Delta q\right)$
5: **if** $ObstacleFree\left(q_{path}\right)$ **then**
6: $c_{new} \leftarrow Cost\left(q_{near}\right) + c\left(q_{rand}\right)$
7: **if** $c_{min} < c_{new}$ **then**
8: $c_{min} \leftarrow c_{new}$
9: $q_{min} \leftarrow q_{new}$
10: **end**
11: **end**
12: **end**
13: **return** q_{min}

Algorithm 3 $T \leftarrow Rewire\left(T, Q_{near}, q_{min}, q_{rand}\right)$

1: **for** $q_{near} \in Q_{near}$ **do**
2: $q_{path} \leftarrow Steer\left(q_{near}, q_{rand}, \Delta q\right)$
3: **if** $ObstacleFree\left(q_{path}\right)$ **and**
$Cost\left(q_{rand}\right) + c\left(q_{path}\right) < Cost\left(q_{near}\right)$ **then**
4: $T \leftarrow ReConnect\left(q_{rand}, q_{near}, T\right)$
5: **end**
6: **return** T

The rewiring algorithm is used to check whether there are nodes around q_{new} (called q_{near}) that will have a shorter distance to q_{start} when connected to q_{new}. If q_{near} has a closer distance to q_{start} if it passes through q_{new}, then q_{new} will be used as the parent of q_{near} (Line 9 in Algorithm 1).

III. THE PROPOSED ALGORITHM: ACS-TSP ALGORITHM

The RRT and ACS hybridization algorithm is shown in Algorithm 4. The RRT-ACS algorithm mimics the character of the ACS algorithm. The ACS metaheuristic is shown in Algorithm 5 [25]. The ACS algorithm consists of three primary functions, which are imitated in the RRT-ACS algorithm. The ConstructSolutions function in the ACS metaheuristic (Lines 4-7 in Algorithm 5) is performed in Lines 6-16 of Algorithm 4. The LocalSearch function (Line 8 in Algorithm 5) is performed in Line 17, and the UpdatePheromones function in the ACS metaheuristic (Line 9 in Algorithm 5) is performed in Line 18 of Algorithm 4.

The hybridization process of the RRT and ACS algorithms is carried out using one of the ACS principles, the pseudorandom proportional rule [44]. A random variable value will be generated at each iteration to determine whether the exploration or exploitation process will be selected. If the exploration process is selected, the determination of the new node will be based on the RRT algorithm. However, if the exploitation process is selected, the determination of the new node will be based on the pheromone distribution information. This process is shown in Lines 8-13 in Algorithm 4.

Algorithm 4 $X^{bs} \leftarrow$ RRT-ACS (map)

```
% ========== Initialization
T ← InitializeTree()
T ← InsertNode (∅, q_init, T)
s ← 0
while termination condition not met do
    for k = 1 to m do
        while s = 0 do
            q ← randvar [0, 1] and τ ≠ ∅
            if q ≤ q_o
                (T, s) ← ACS (T, τ, map)
            else
                (T, s) ← RRT(T, map)
            end if
        end while
        X (k) ← MakePath from T
    end for
    X^bs ← LocalSearch (X^bs, map)
    τ ← UpdatePheromones (X, τ)
end while
```

Algorithm 5 ACSMetaheuristic

```
Initialize
Loop
    Each ant is positioned on a starting node
    Loop /* ConstructsSolutions */
        Each ant applies a state transition rule to
        Incrementally build a solution
        Each ant applies a local pheromone
        Updating rule
    Until all ants have built a complete solution
    Each ant is brought to a local minimum
     using a tour improvement heuristic
     /* LocalSearch */
    A global pheromone updating rule is applied
     /* UpdatePheromones */
Until EndCondition
```

Algorithm 6 $(T, s) \leftarrow \boldsymbol{RRT}(T, map)$

```
1  q_rand ← RandomSample (k)
2  Q_near ← Near (T, q_rand)
3  q_parent ← NearestNeighbor (q_rand, Q_near, T)
4  q_new ← Steer (q_nearest, q_rand, Δq)
5  if Obstaclefree (q_new, q_nearest, map) then
6    Q_near ← Near (T, q_new)
7    q_min ← ChooseParent (q_new, Q_near, q_nearest)
8    T ← InsertNode (q_min, q_new, T)
9    if d (q_new, q_goal) ≤ Δq and Obstaclefree then
10     T ← InsertNode (q_goal q_new, T)
11     s ← 1
13   end if
14 end if
```

Algorithm 7 $(T, s) \leftarrow \boldsymbol{ACS}(T, map)$

```
1  q_samp ← RandomSampleFrom10LastNode (T)
2  τ_near ← Near (τ, q_samp)
3  q_rand ← RouletteWhell (τ_near)
4  advanced = 0
5  while s = 0 and advanced = 0 do
6    Q_near ← Near (T, q_rand)
7    q_parent ← NearestNeighbor (q_rand, Q_near, T)
8    q_new ← Steer (q_nearest, q_rand, Δq)
9    if Obstaclefree (q_new, q_nearest, map) then
10     Q_near ← Near (T, q_new)
11     q_min ← ChooseParent (q_new, Q_near, q_nearest)
12     T ← InsertNode (q_min, q_new, T)
13     if d (q_new, q_goal) ≤ Δq and Obstaclefree then
14       T ← InsertNode (q_goal q_new, T)
15       s ← 1
16     end if
17   else
18     advanced = 1
19   end if
20   if q_new = q_rand then advanced = 1
21 end while
```

If the exploration process is selected, then the determination of the new node in this iteration will be carried out based on the RRT algorithm, whose detailed algorithm is shown in Algorithm 6 [12]. On the other hand, if the exploitation process is selected, then the determination of the new node in this iteration will be carried out based on the ACS algorithm, whose detailed algorithm is shown in Algorithm 7. There are two main differences between Algorithm 6 and Algorithm 7. The first difference is in terms of sampling. The RRT algorithm performs this sampling process randomly using the RandomSample function (Line 1 in Algorithm 6).

In contrast, the ACS algorithm performs this sampling process based on information from the distribution of pheromones. Information from the distribution of pheromones will show which areas have a higher chance of producing the optimal pathway. The sampling function based on this pheromone information is shown in Lines 1-3 in Algorithm 7.

Another difference between the RRT and ACS algorithms is the tree extension. The tree extension is carried out in only one step in length (Δq) in the RRT algorithm. In the ACS algorithm, tree extension is carried out in many steps, not just one step in length. Each time the nearest neighbor has been selected, the tree will grow toward the x_{rand} until it is reached or an obstacle prevents growth. This new node extension process is shown in Lines 4, 5, 18, and 20 in Algorithm 7. This process mimics RRT/RRT*-connect [20]–[22]. An illustration of this process is shown in Fig. 1.

After each ant has carried out the path planning process, the next process in the ACS algorithm is to perform LocalSearch, whose detailed algorithm is shown in Algorithm 8. This LocalSearch process mimics the behavior of the informed

Algorithm 8 $X_{sol(new)} \leftarrow$ ***LocalSearch***(X_{sol}, map)

1: $T \leftarrow InitializeTree()$
2: $T \leftarrow InsertNode\,(\emptyset, q_{init}, T)$
3: **While** *termination condition not met* **do**
4: $q_{rand} \leftarrow SamplingNearBestPath\,(X_{sol}, d)$
5: $Q_{near} \leftarrow Near\,(T, q_{rand})$
6: $q_{parent} \leftarrow NearestNeighbor\,(q_{rand}, Q_{near}, T)$
7: $q_{new} \leftarrow Steer\,(q_{nearest}, q_{rand}, \Delta q)$
8: **if** $Obstaclefree\,(q_{new}, q_{nearest}, map)$ **then**
9: $Q_{near} \leftarrow Near\,(T, q_{new})$
10: $q_{min} \leftarrow ChooseParent\,(q_{new}, Q_{near}, q_{nearest})$
11: $T \leftarrow InsertNode\,(q_{min}, q_{new}, T)$
12: **if** $d\,(q_{new}, q_{goal}) \leq \Delta q$ **and** $Obstaclefree$ **then**
13: $T \leftarrow InsertNode\,(q_{goal} q_{new}, T)$
14: $s \leftarrow 1$
15: **end if**
16: **end if**
17: **end while**

Algorithm 9 $\tau \leftarrow$ ***UpdatePheromone*** (X, τ)

1: $X^w \leftarrow$ Select w best $Path\,(X)$
2: $\tau \leftarrow$ Add $Pheromone\ Node$ on the w best $Path\,(X^w, \tau)$
3: $\tau \leftarrow$ Evaporate $Pheromone\ Node\,(\tau)$

RRT* algorithm, with modifications to the sampling process shown in Line 4 of Algorithm 8. The sampling process is carried out around the current best path (X^{bs}) to obtain a better path.

After carrying out the LocalSearch process, the next process in the ACS algorithm is to perform an UpdatePheromone whose algorithm details are shown in Algorithm 9. To provide strong additional reinforcement only to the good paths, only w ants with the best path will provide additional pheromones at each iteration. This process mimics the rank-based ant system algorithm proposed by Bullnheimer *et al.* [45]. This process is shown in Line 1 of Algorithm 9.

On each path, there are several interconnected nodes. The distance from each node is Δq. If a path belongs to the w best path, several pheromones will be deposited on the nodes of that path. The pheromone value (τ) deposited on that path will be inversely proportional to the length of the path. As a result, the better/shorter the resulting path, the greater is the value of the pheromone deposited. For example, using the case of benchmark II (which will be explained in Table 2), the number of pheromones deposited can be illustrated as shown in Fig. 2. There are several paths constructed by several ants as shown in Fig. 2. Each complete path connecting the start node to the end node is illustrated using a different color line. The green line gives an example of a long-distance path. Using the formula $\tau = Q/PathLength$, the pheromone value (τ) obtained for the green line is 227.6. Meanwhile, the red line gives an example of a shorter distance path. The pheromone value on the red line is 245.1. This illustrates the principle of determining the pheromone value (τ) in the

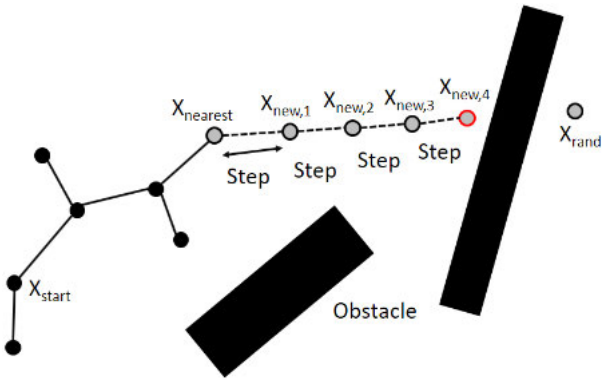


FIGURE 1. In the RRT-ACS algorithm, while performing the exploitation process, the tree is extended by many steps at each iteration.

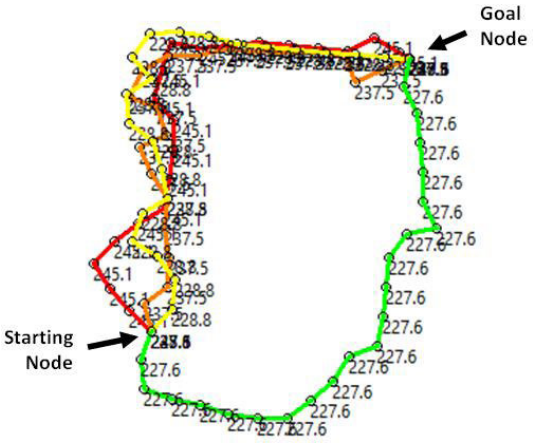


FIGURE 2. Illustration of the pheromone value comparison of paths with different lengths.

RRT-ACS algorithm, that is, that the pheromone value is greater in the shortest path.

In the ACS algorithm, the probability of choosing a node is influenced not only by the pheromone value (τ), but also by the heuristic information value (η) of the node. In the RRT-ACS algorithm, the heuristic information value (η) is the distance from a node to the destination node. Then the heuristic information value of a node will be greater if the distance is closer to the destination node. The illustration is shown in Fig. 3. The definition of this heuristic information differs from the standard ant algorithm. This modification is based on the fact that not all nodes that have the pheromone must be visited. There is no urgency to visit the nearest node first. An illustration of this heuristic information can be seen in the blue line in Fig. 3. The blue line consists of several nodes. Nodes closer to the goal node have a greater heuristic information value than nodes that are farther away from the goal node.

IV. SIMULATION RESULTS AND DISCUSSION

A. SIMULATIONS SETUP

Some simulations were carried out to demonstrate the performance of the proposed ACS-TSP algorithm in the path planning problem. All simulations and analyses were conducted

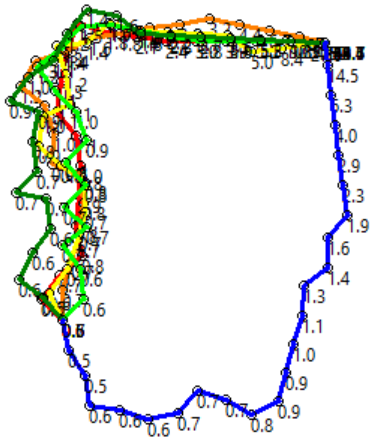


FIGURE 3. Illustration of the heuristic information values. The heuristic information value of a node will be greater if the distance is closer to the destination node.

using the same PC (with a Core i5 3.20 GHz CPU and 4 GB RAM) with Windows 10 64-bit Enterprise for a reliable comparison. The LabVIEW 7.1 programming language is used to implement the proposed algorithm and other algorithms in the comparative study.

B. PARAMETERS SETTINGS

In the ant colony algorithm, there are several parameters, such as the number of ants (m), parameters that determine the relative influence of the pheromone trail (α), parameters that determine the heuristic information (β), the pheromone evaporation rate (ρ), the number of the best ants that contributed to the addition of the pheromone (w), and the total pheromone number (Q). The parameter values for the ACS are similar to the settings recommended in the literature [25]. Several parameters in the RRT algorithm and the RRT* algorithm (and RRT* variations) are *the expanded distance*, *the probability of sampling the goal node*, and γ. All parameters used in this simulation are shown in Table 1.

TABLE 1. Parameter settings used in the simulation.

Parameter	Value
m	10
α	1
β	2
ρ	0.1
Q	100
q_o	0.9
w	6
Expand distance	20
The probability of sampling the goal node	5%
γ	50

C. COMPARISON OF THE CONVERGENCE RATE AND OPTIMALITY

To test the proposed algorithm's convergence speed and optimality performance, the RRT-ACS algorithm is compared with the RRT*, RRT*-connect, informed RRT*, and informed RRT*-connect algorithms. The test was performed using ten benchmark cases found in the RRT*, RRT*-connect, informed RRT*, informed RRT*-connect, and RRT*-smart algorithm publications. The benchmark cases used are presented in Tables 2 and 3.

Furthermore, all algorithms are run 100 times, each with 20,000 iterations. The result obtained is that there are

TABLE 2. Benchmark case used in simulation testing.

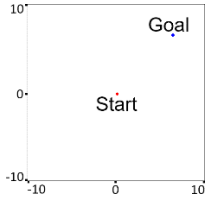
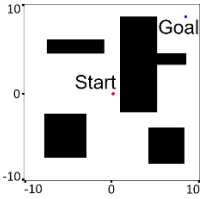
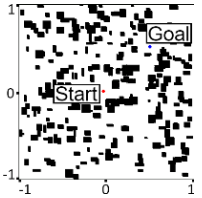
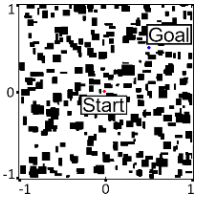
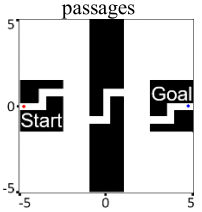
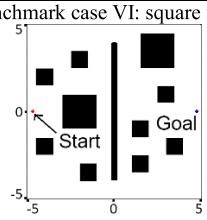
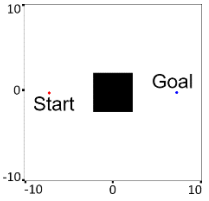
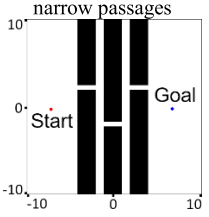
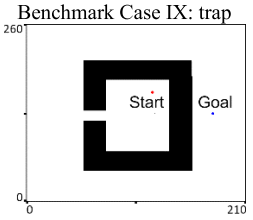
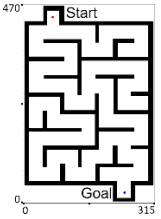
Benchmark Name and Map	Description
Benchmark case I: no obstacle 	This benchmark case was used by Karaman [14] when publishing the RRT* algorithm. This environment is used to indicate whether an algorithm can produce a convergent path to the optimal or not.
Benchmark case II: simple obstacle 	This benchmark case was used by Karaman [14] when publishing the RRT* algorithm. This environment is used to indicate whether an algorithm can produce a convergent path to the optimal or not
Benchmark case III: clutter I 	This benchmark case was used by Gammel et al. [10] when publishing the informed RRT* algorithm. This environment is used to test the convergence speed of the path planning algorithm.
Benchmark case IV: clutter II 	This benchmark case was used by Gammel et al. [10] when publishing the informed RRT* algorithm. This environment is used to test the convergence speed of the path planning algorithm.
Benchmark case V: tough passages 	This benchmark case was used by Klem et al. [22] when publishing the RRT* -connect algorithm. This environment is used to test the effectiveness of the path planning algorithm when the goal node is hidden behind a narrow passage.

TABLE 3. Benchmark case used in simulation testing.

Benchmark Name and Map	Description
Benchmark case VI: square field 	This benchmark case was used by Klem et al. [22] when publishing the RRT* -connect algorithm.
Benchmark case VII: cube obstacle 	This benchmark case was used by Mashayekhi et al. [24] when publishing the informed RRT*-connect algorithm. This environment is used to test the effectiveness of the path planning algorithm
Benchmark case VIII: multiple narrow passages 	This benchmark case was used by Mashayekhi et al. [24] when publishing the informed RRT*-connect algorithm. This environment is used to test the effectiveness of the path planning algorithm when the goal node is hidden behind a narrow gap.
Benchmark Case IX: trap 	This benchmark case was used by Islam et al. [46] when publishing the RRT*-smart algorithm
Benchmark Case X: maze 	This benchmark case was used by Nasir et al. [42]

100 search trees for each algorithm. Then the best path value from each search tree will be averaged for each iteration.

The test results on benchmark case I are shown in Fig. 4. The best convergence rate are presented by the informed RRT*-connect and RRT-ACS algorithms. The informed RRT*, informed RRT*-connect, and RRT-ACS algorithms achieved the optimal values. The RRT* and RRT*-connect algorithms still need time to reach their optimal values. The informed RRT*-connect and RRT-ACS algorithms can quickly reach optimal values by limiting the sampling area.

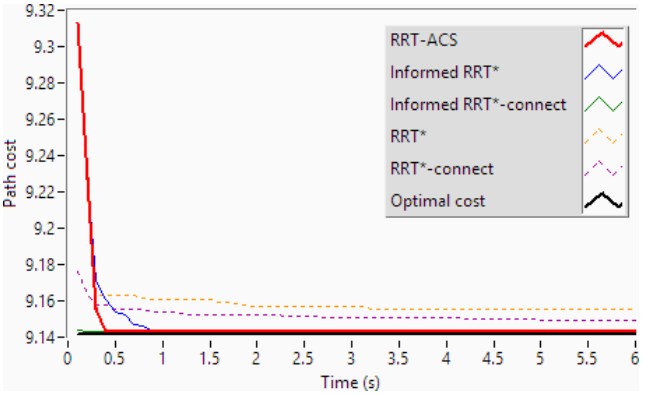


FIGURE 4. The test results on benchmark case I: no obstacle. The cost of the best paths plotted against time-averaged over 100 trials.

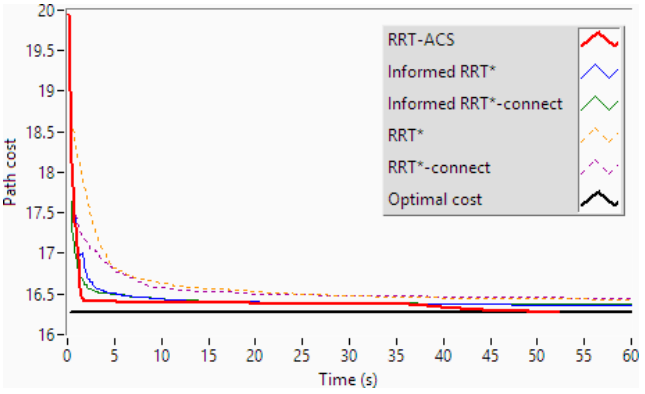


FIGURE 5. The test results on benchmark case II: simple obstacle. The cost of the best paths plotted against time-averaged over 100 trials.

The test results on benchmark case II are shown in Fig. 5. The best convergence rate is the RRT-ACS algorithm, followed by informed RRT*-connect and informed RRT*. The graph shows that the RRT-ACS algorithm achieved the optimal values. However, the path cost of the RRT*, informed RRT*, RRT*-connect, and informed RRT*-connect algorithms still appear to be decreasing, indicating that the four algorithms still need time to reach the values that the RRT-ACS algorithm has achieved.

The test results on benchmark case III are shown in Fig. 6. The best convergence rate is the informed RRT*-connect and informed RRT* algorithms. In addition, the informed RRT*, informed RRT*-connect, and RRT-ACS algorithms succeeded in achieving the shortest path cost compared to those of the RRT* and RRT*-connect algorithms.

The test results on benchmark case IV are shown in Fig. 7. The best convergence rate is the informed RRT*-connect algorithm. In addition, the informed RRT*, informed RRT*-connect, and RRT-ACS algorithms succeeded in achieving the shortest path cost compared to those of other existing algorithms.

The test results on benchmark case V are shown in Fig. 8. Again, the best convergence rate is the RRT-ACS algorithm.

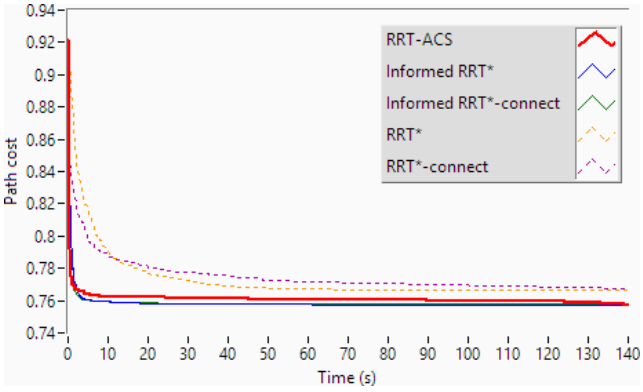


FIGURE 6. The test results on benchmark case III: clutter. The cost of the best paths plotted against time-averaged over 100 trials.

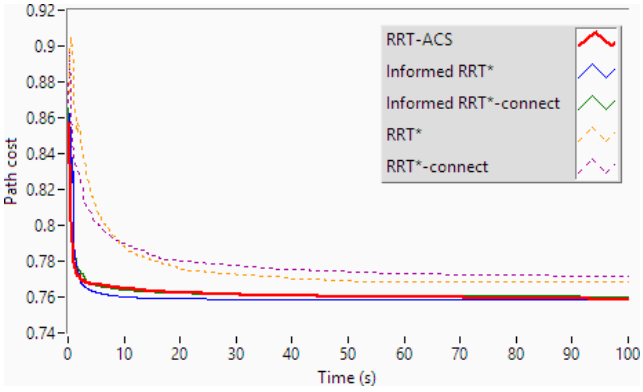


FIGURE 7. The test results on benchmark case IV: clutter. The cost of the best paths plotted against time-averaged over 100 trials.

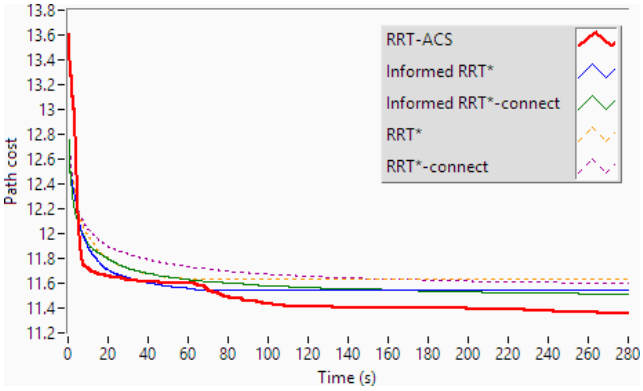


FIGURE 8. The test results on benchmark case V: toughpassages. The cost of the best paths plotted against time-averaged over 100 trials.

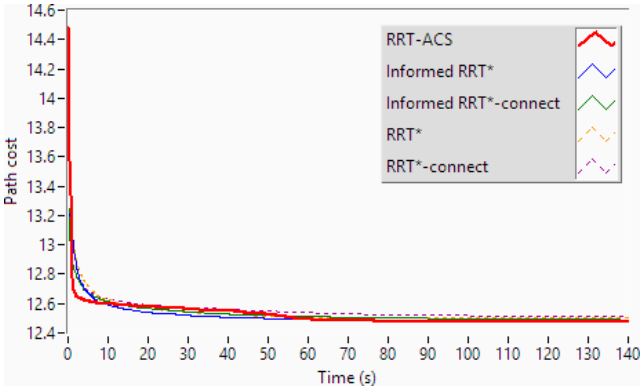


FIGURE 9. The test results on benchmark case VI: squarefield. The cost of the best paths plotted against time-averaged over 100 trials.

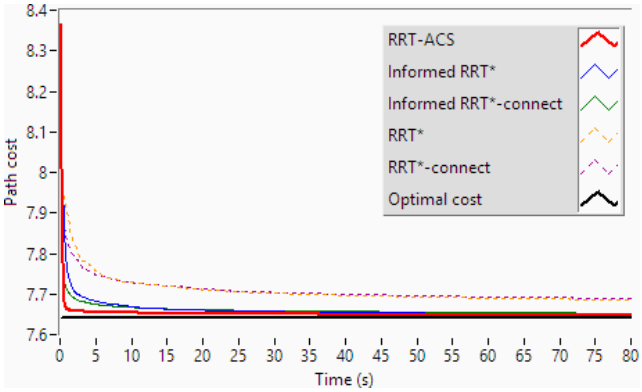


FIGURE 10. The test results on benchmark case VII: cube obstacle. The cost of the best paths plotted against time-averaged over 100 trials.

In addition, the RRT-ACS algorithm succeeded in achieving the shortset path cost compared to other existing algorithms.

The test results on benchmark case VI are shown in Fig. 9. The best convergence rate is the informed RRT* and informed RRT*-connect algorithms. In addition, the five tested algorithms succeeded in achieving the shortest path length.

The test results on benchmark case VII are shown in Fig. 10. The best convergence rate is the RRT-ACS algorithm. The informed RRT*, informed RRT*-connect and RRT-ACS algorithms succeeded in achieving optimal values. The RRT* and RRT*-connect algorithms still need time to reach their optimal values.

The test results on benchmark case VIII are shown in Fig. 11. The best convergence rate is the RRT-ACS algorithm. The RRT-ACS algorithm succeeded in producing the path with the shortest length. The informed RRT*-connect, informed RRT*, RRT*-connect, and RRT* algorithms still need time to reach optimal values.

The test results on the benchmark case IX are shown in Fig. 12. The best convergence rate is the RRT-ACS algorithm. The test results on the benchmark case X are shown in Fig. 13. The best convergence rate is the RRT-ACS algorithm.

The results of this test indicate that the RRT-ACS algorithm has a good convergence speed. From Fig. 4, Fig. 5, and Fig. 10 – Fig. 13, for benchmark cases with a known optimal value, it is observed that the proposed algorithm has succeeded in achieving this optimal value.

Tables 4 and 5 present the statistical results of the performance measurements obtained by RRT-ACS and other existing algorithms in different benchmark cases. The numbers in the tables represent the path lengths. Performance

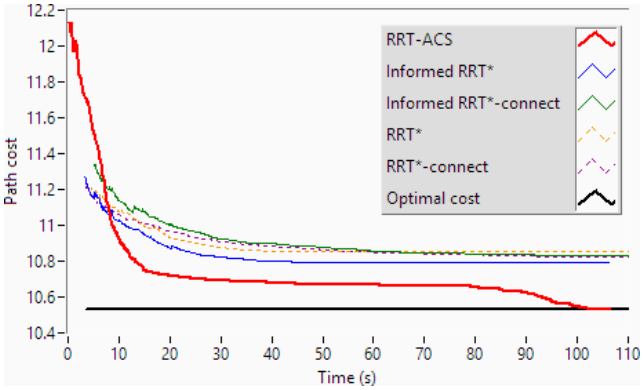


FIGURE 11. The test results on benchmark case VIII: multiple narrow passages. The cost of the best paths plotted against time-averaged over 100 trials.

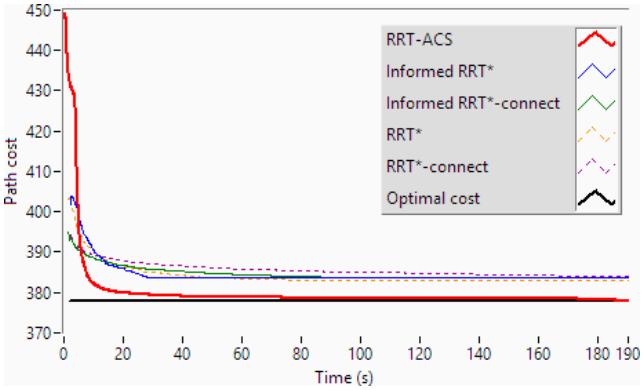


FIGURE 12. The test results on benchmark case IX: trap. The cost of the best paths plotted against time-averaged over 100 trials.

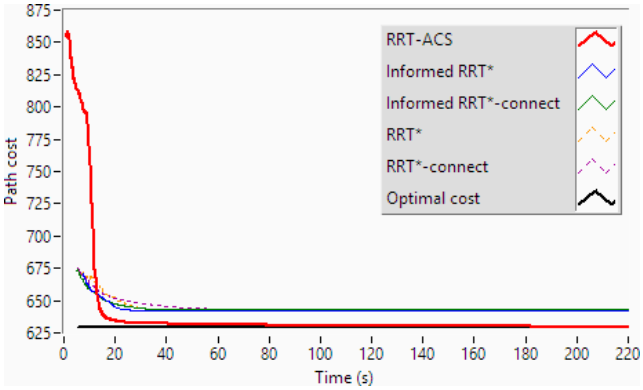


FIGURE 13. The test results on benchmark case X: maze. The cost of the best paths plotted against time-averaged over 100 trials.

measurements include the best path length, the worst path length, the mean path length, and the standard deviation.

It can be seen from Tables 4 and 5 that the algorithm that has the best result for each section is between informed RRT*, informed RRT*-connect, and RRT-ACS. Therefore, it is concluded that the RRT-ACS algorithm has good performance compared to other existing algorithms for the best path length, worst path length, mean path length, and standard deviation of the path length.

D. STATISTICAL ANALYSIS OF THE SIMULATION RESULTS

Derrac *et al.* [47] and Zhang *et al.* [30] proposed that statistical analysis should be carried out to improve the performance evaluation in various algorithms. Therefore, following Zhang's paper, two statistical analyses were carried out based on the results obtained in Tables 4 and 5. The statistical software package SPSS was used for this analysis.

The first statistical test is based on the Friedman nonparametric test. Friedman's nonparametric test was used to verify whether there was a significant difference in performance among all algorithms. The average ranking obtained by this nonparametric Friedman test for each algorithm is shown in Table 6. The algorithm performs better if it has a lower average ranking. The Friedman statistic value obtained in

TABLE 4. Statistical results obtained by each algorithm in different benchmark cases. The best result for each section is highlighted in bold.

Environments	Algorithm	Best	Worst	Mean	Std
Benchmark Case I	RRT-ACS	**9.1429**	**9.1429**	**9.1429**	**0.0000**
	Informed RRT*	**9.1429**	**9.1429**	**9.1429**	**0.0000**
	Informed RRT*-connect	**9.1429**	**9.1429**	**9.1429**	**0.0000**
	RRT*	**9.1429**	9.1825	9.1519	0.0085
	RRT*-connect	**9.1429**	9.1538	9.1462	0.0033
Benchmark Case II	RRT-ACS	**16.2703**	**16.3420**	**16.2750**	**0.0137**
	Informed RRT*	16.2736	16.4514	16.3519	0.0429
	Informed RRT*-connect	16.2726	16.4189	16.3316	0.0373
	RRT*	16.3241	16.6236	16.4344	0.0689
	RRT*-connect	16.2929	16.5802	16.4052	0.0675
Benchmark Case III	RRT-ACS	**0.7571**	0.7581	0.7573	0.0002
	Informed RRT*	**0.7571**	**0.7576**	**0.7572**	**0.0001**
	Informed RRT*-connect	**0.7571**	0.7578	0.7573	0.0002
	RRT*	0.7594	0.7970	0.7669	0.0062
	RRT*-connect	0.7586	0.8066	0.7649	0.0064
Benchmark Case IV	RRT-ACS	0.7579	0.7602	0.7581	0.0004
	Informed RRT*	**0.7577**	**0.7590**	**0.7580**	0.0003
	Informed RRT*-connect	0.7594	0.7600	0.7596	**0.0002**
	RRT*	0.7612	0.7955	0.7687	0.0060
	RRT*-connect	0.7621	0.7820	0.7689	0.0037

TABLE 5. Statistical results obtained by each algorithm in different benchmark cases. The best result for each section is highlighted in bold.

Environments	Algorithm	Best	Worst	Mean	Std
Benchmark Case V	RRT-ACS	**11.3204**	**11.5479**	**11.3542**	**0.032**
	Informed RRT*	11.4503	11.7476	11.5390	0.053
	Informed RRT*-connect	11.4275	11.6050	11.5086	0.037
	RRT*	11.4754	11.8306	11.6332	0.074
	RRT*-connect	11.4649	11.7586	11.5976	0.057
Benchmark Case VI	RRT-ACS	**12.3961**	12.7550	12.5010	0.105
	Informed RRT*	12.4401	**12.5804**	12.4914	0.030
	Informed RRT*-connect	12.4280	12.5411	**12.4741**	**0.024**
	RRT*	12.4414	12.5945	12.5042	0.029
	RRT*-connect	12.4366	12.5466	12.4911	0.027
Benchmark Case VII	RRT-ACS	**7.6407**	7.6598	**7.6442**	0.007
	Informed RRT*	7.6409	**7.6527**	7.6468	**0.003**
	Informed RRT*-connect	7.6409	7.6544	7.6468	**0.003**
	RRT*	7.6565	7.7386	7.6866	0.016
	RRT*-connect	7.6504	7.7073	7.6755	0.012
Benchmark Case VIII	RRT-ACS	**10.5243**	**10.5513**	**10.5278**	**0.004**
	Informed RRT*	10.6572	11.3230	10.7890	0.094
	Informed RRT*-connect	10.6850	11.3658	10.8283	0.121
	RRT*	10.6762	11.5968	10.8505	0.118
	RRT*-connect	10.6719	11.0543	10.8222	0.079
Benchmark Case IX	RRT-ACS	**378.25**	**380.71**	**378.92**	**0.709**
	Informed RRT*	379.84	387.92	383.62	2.063
	Informed RRT*-connect	379.62	387.57	383.82	1.740
	RRT*	382.05	387.80	383.10	1.878
	RRT*-connect	382.13	385.23	383.66	0.967
Benchmark Case X	RRT-ACS	**616.77**	**632.54**	**628.23**	4.651
	Informed RRT*	630.69	658.86	642.43	5.650
	Informed RRT*-connect	630.66	661.90	642.80	6.394
	RRT*	631.03	652.79	641.87	4.188
	RRT*-connect	633.86	650.60	643.08	**3.527**

this test is 21.18. Given that the confidence interval has been set at 95%, this gives the critical point of 11,07. Thus, since the Friedman statistic value of 21.18 > 11.07, we can say that there is a statistically significant difference between the algorithms based on the average rank returned by the Friedman nonparametric test. Table 6 shows that the RRT-ACS algorithm can be considered the best method with the lowest ranking.

TABLE 6. The average ranking for each algorithm obtained by the Friedman test.

Algorithms	Average ranking
RRT-ACS	1.7
Informed RRT*	2.35
Informed RRT*-Connect	2.75
RRT*	4.3
RRT*-Connect	3.9

However, the Friedman test can only detect significant differences in the overall algorithms being compared. Moreover, Friedman's test could not establish an exact class in particular that differs from each other [47]. Therefore, the second statistical test is an analysis of variance (ANOVA) with the p values adjusted by the Bonferroni correction. Table 7 shows the adjusted significance value between the RRT-ACS and the other algorithms.

The RRT-ACS algorithm is significantly better than the other algorithms at the 95% confidence level if the adjusted significant values are lower than 0.05. From Table 7, it can be seen that the adj. sig. values between RRT-ACS and RRT* and RRT*-Connect are lower than 0.05 (except for one benchmark). Moreover, more adj. sig. values between RRT-ACS and informed RRT* and informed RRT*-connect are higher than 0.05. Therefore, it can be concluded that based on statistical tests, the RRT-ACS algorithm has advantages over the RRT* and RRT*-connect algorithms at a 95% confidence level.

Based on Fig. 4 – Fig. 13, and Table 7, the RRT-ACS algorithm only shows convergence speed and path cost values that compete with the informed RRT*-connect and informed RRT* algorithms in the no obstacle, simple obstacle, clutter, square field, and cube obstacle cases. However, the RRT-ACS algorithm shows an advantage over the other algorithms at the 95% confidence level on tough passages, multiple narrow passages, maze, and trap cases.

The advantage of the informed RRT*-connect and informed RRT* algorithms is the ability to limit the sampling area to an ellipsoid subset of the state space whose eccentricity depends on the length of the shortest current solution. By limiting the sampling area, the informed RRT* algorithm and the informed RRT*-connect algorithm can produce shorter paths, as discussed by Gammell *et al.* [10], Mashayekhi *et al.* [24], and Wang *et al.* [23].

However, in the tough passages, multiple narrow passages, maze, and trap cases, the length of the shortest current solution is too long so that the resulting ellipsoid sampling area has a larger size than the search environment, as shown in Fig. 14. Therefore, the resulting ellipsoid area, in this case, cannot contribute to limiting the sampling area. Therefore, the performance of the informed RRT* algorithm will be similar to the performance of the RRT* algorithm. The performance of the informed RRT*-connect algorithm will be similar to the performance of the RRT*-connect algorithm, as shown in Fig. 8 and Fig. 11 - Fig. 13. Kim and Song [48] also

TABLE 7. The adjusted significance values between the RRT-ACS and other existing algorithms.

Environments	RRT-ACS vs.	Std. Error	Adj. Sig
Benchmark Case I	Informed RRT*	0.012	1
	Informed RRT*-Connect	0.012	1
	RRT*	0.014	0.00
	RRT*-Connect	0.014	0.00
Benchmark Case II	Informed RRT*	0.238	0.40
	Informed RRT*-Connect	0.238	0.40
	RRT*	0.206	0.00
	RRT*-Connect	0.206	0.00
Benchmark Case III	Informed RRT*	0.164	1
	Informed RRT*-Connect	0.175	1
	RRT*	0.151	0.00
	RRT*-Connect	0.151	0.00
Benchmark Case IV	Informed RRT*	0.141	1
	Informed RRT*-Connect	0.142	0.48
	RRT*	0.122	0.00
	RRT*-Connect	0.122	0.00
Benchmark Case V	Informed RRT*	0.373	0.00
	Informed RRT*-Connect	0.431	0.00
	RRT*	0.373	0.00
	RRT*-Connect	0.373	0.00
Benchmark Case VI	Informed RRT*	0.380	1
	Informed RRT*-Connect	0.380	0.07
	RRT*	0.329	1
	RRT*-Connect	0.382	1
Benchmark Case VII	Informed RRT*	0.097	1
	Informed RRT*-Connect	0.097	1
	RRT*	0.084	0.00
	RRT*-Connect	0.084	0.00
Benchmark Case VIII	Informed RRT*	0.627	0.00
	Informed RRT*-Connect	0.726	0.00
	RRT*	0.628	0.00
	RRT*-Connect	0.672	0.00
Benchmark Case IX	Informed RRT*	0.904	0.00
	Informed RRT*-Connect	0.904	0.00
	RRT*	0.904	0.00
	RRT*-Connect	1.044	0.00
Benchmark Case X	Informed RRT*	1.490	0.00
	Informed RRT*-Connect	1.490	0.00
	RRT*	1.490	0.00
	RRT*-Connect	1.490	0.00

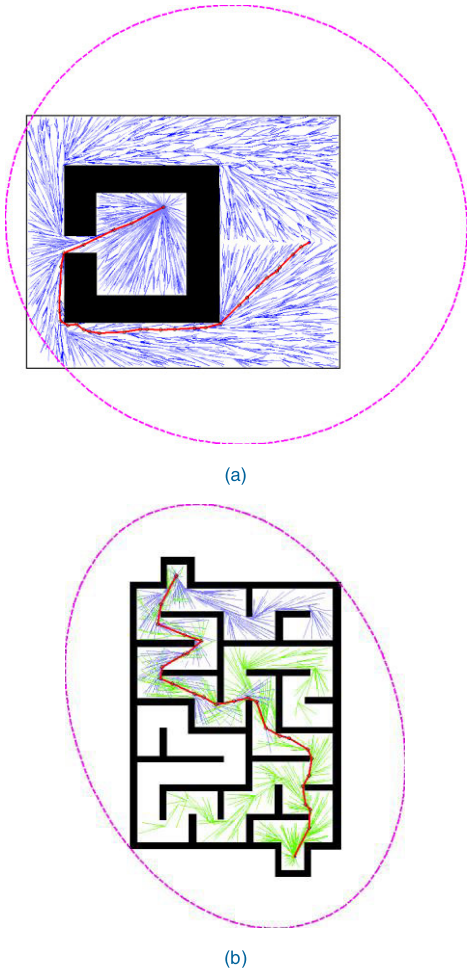


FIGURE 14. The length of the shortest current solution is too long so that the resulting ellipsoid sampling area in the informed RRT* and informed RRT*-connect has a larger size than the search environment: (a) Informed RRT* in the trap case, (b) Informed RRT*-connect in the maze case.

reported the same weakness of this informed RRT* algorithm. The RRT-ACS algorithm, which uses pheromone information from the previous search, does not have such weaknesses and can produce better paths in the maze case and trap case.

E. THE EFFECT OF THE EXPLORATION AND EXPLOITATION PARAMETER (q_0)

As previously stated, the hybridization process of the RRT and ACS algorithms is performed using one of the ACS principles, that is, the pseudorandom proportional rule. At each iteration, a random variable value (q) will be generated to determine whether to execute the exploration or exploitation process. Then a certain number (q_0) will be selected as a threshold. If $q \leq q_o$, then the iteration will execute the exploration process. However, if the value $q \geq q_0$, then the iteration process will be exploited. If the exploration process is selected, the determination of the new node will be based on the RRT algorithm. However, if the exploitation process is selected, the determination of the new node will be based on the pheromone distribution information.

Then in this section, the test results regarding the effect of system performance on the varying values of q_0 will be presented. The test is carried out on several benchmark cases: benchmark cases I, II, and III. The sampling interval of q_0 is 0.1. The value of q0 can be determined to be any value as long as it is in the interval 0 to 1 [25]. Various research publications report the use of q0 with different values. Mavrovouniotis *et al.* [49] used the value of q0 $\in$ {0.0, 0.2, 0.5, 0.9}. Zhang *et al.* [30] used the value of q0 $\in$ {0.4, 0.5, 0.6, 0.7, 0.8, 0.9]. By following the examples from Mavrovouniotis and Zhang, this study used a sampling interval of q0 with a value of 0.1.

The test results can be seen in Fig. 15 - Fig. 17. At each value of q_0, the RRT-ACS algorithm is run 100 times, each until it is close to its optimal value with a tolerance of 1%.

From the test results, it can be seen that when q_0 is 0.1 to 0.3, the computation time required by the algorithm to produce a path, with a tolerance of 1% of its optimal value,

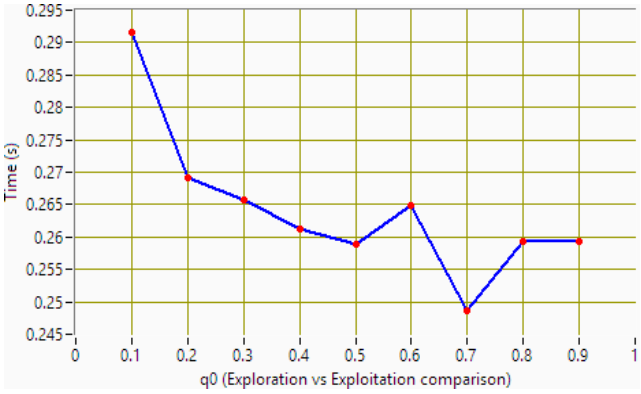


FIGURE 15. The effect of q_0 variation on system performance. The test results are for benchmark case I.

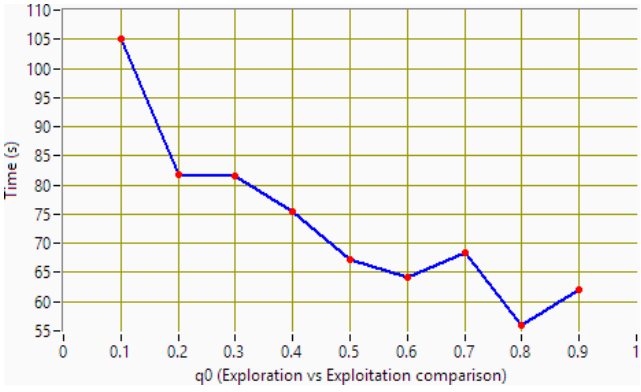


FIGURE 16. The effect of q_0 variation on system performance. The test results are for benchmark case II.

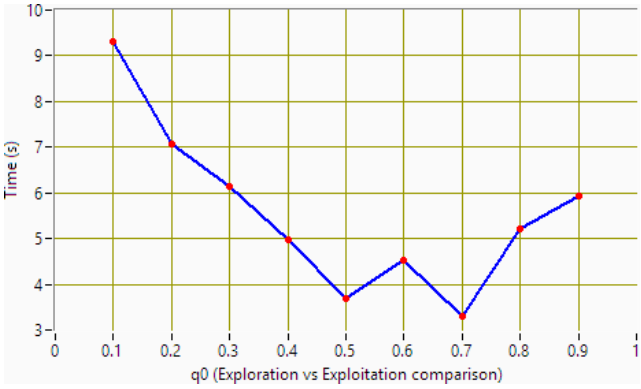


FIGURE 17. The effect of q_0 variation on system performance. The test results are for benchmark case III.

is the longest. This is because the behavior of the algorithm is similar to that of the RRT algorithm. However, when the search role based on pheromone information increases (q_0 is more than 0.4), the required computation time is decreased.

From the three test results, the fastest average computation time is obtained when q_0 is 0.5. If q_0 is more than 0.5, there are cases where the computation time of the algorithm is faster. However, there are also cases where the computation time of the algorithm is increased. The test results show that the computation time is increased when the q_0 value is more than 0.5. However, that computation time is still faster than if the q_0 value is between 0.1 and 0.3.

F. THE STABILITY AND ROBUSTNESS OF THE RRT-ACS ALGORITHM

According to Xue *et al.* [50], a path planning algorithm is stable if the same paths are generated repeatedly when planning the same task. Therefore, we will use the data in Tables 4 and 5 to analyze the stability of the RRT-ACS algorithm.

Tables 4 and 5 present the statistical results of the performance measurements obtained by RRT-ACS and other existing algorithms in different benchmark cases. The numbers in the tables represent the path lengths. The performance measurements include the best path length, the worst path length, the mean path length, and the standard deviation. If the standard deviation of an algorithm decreases, then the path cost values generated in each iteration are more similar.

The table shows that in each benchmark case test, the standard deviation of the RRT-ACS algorithm is the smallest or relatively small compared to those of other algorithms. This small standard deviation value shows that the RRT-ACS algorithm can be declared stable compared to other existing algorithms.

According to Luan and Fang [51], the robustness of the path planning algorithm can be proven by testing the algorithm in various environments. Therefore, we tested the RRT-ACS algorithm on ten benchmark cases, where the results are shown in Fig. 4 – Fig. 13. From those pictures, we can see that the final path can be found, proving the RRT-ACS algorithm's robustness.

G. THE CONVERGENCE OF THE RRT-ACS ALGORITHMS

According to La Valle [52], a path planning algorithm is said to be converge if the algorithm can achieve a stationary cost-to-go function because the values of the final path no longer change. However, it is not enough for an algorithm to only have convergence properties in the path planning problem. A good path planning algorithm must converge to an optimal value, that is, it must possess an optimal asymptotic property [1].

According to Karaman and Frazzoli [14], a path planning algorithm can be proven to be asymptotic optimally through theoretical proof or experimental testing. Therefore, this paper conducted an experimental test to show that the RRT-ACS algorithm can converge to the optimal value. The test results from benchmark case tests with known optimal values (cases I, II, VII - X) indicate that the proposed algorithm has succeeded in achieving those optimal values. Although proof analysis is not presented in this paper, based on the testing on various path planning cases that we performed, our proposed RRT-ACS algorithm successfully achieved those optimal values.

H. THE RAPIDITY OF THE RRT-ACS ALGORITHMS

The RRT-ACS algorithm is a hybridization of the RRT algorithm and the ACS algorithm. The RRT and RRT* algorithms have rapid exploration ability properties [41]. The rapid exploration of RRT and RRT* algorithms is achieved by taking a random sample from the free configuration space and extending the tree in the direction of the random sample [1]. In this way, the RRT and RRT* algorithms can rapidly explore the state space and quickly connect every new state to the explored state [53]. The RRT-ACS algorithm also extends the search tree in the direction of the random sample or the direction of the previous pheromone information. Therefore, the RRT-ACS algorithm still has the rapid exploration property that the RRT and RRT* algorithms have.

We also tested ten different benchmarks to show that the rapidity of the RRT-ACS algorithm to reach the optimal value is better than the rapidity of the RRT* algorithm to reach the optimal value. In the test results shown in Fig. 4 – Fig. 13, the convergence speed of the RRT-ACS algorithm toward the optimal value is better than the convergence speed of the RRT* algorithm. Therefore, we can conclude that the RRT-ACS algorithm has a good rapidity property.

I. THE COMPARISON OF THE RRT-ACS ALGORITHM WITH ACO AND RELATED IMPROVED ALGORITHMS

We compared the RRT-ACS algorithms with the classical ACO algorithm (ant system) [27], two variants of the ACO: (rank-based ant system (RAS) [45] and ACS [25]), and two state-of-the-art optimization algorithms based on the ACO: PS-ACO [54] and AIACSE [30]. The common parameters used for the above ant system algorithms are shown in Table 8. The parameter values of PS-ACO are identical to those in [54].

TABLE 8. Parameter settings used in the ACO, RAS, ACS, and AIACSE simulations.

Parameter	Value
Number of ants	2*width
α	1.1
β	7
ρ	0.2
ζ	0.2
Q	100
q_o (ACS)	0.8
$q_o min$ (AIACSE)	0.4
$q_o max$ (AIACSE)	0.9
NC_{max}	200

This experiment is conducted under four different size environments, as shown in Fig. 18. The optimal path of each grid map generated by RRT-ACS and AIACSE is depicted in Fig. 18 and Fig. 19. Table 9 provides the statistical results of the performance metrics obtained by RRT-ACS and other existing ant algorithms under different maps. The numbers in the tables represent the path lengths. Performance

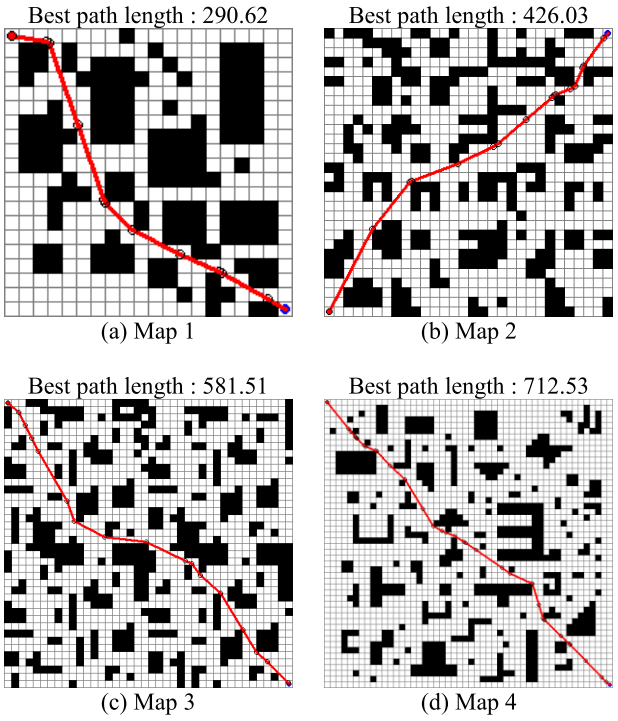


FIGURE 18. The best path generated by the RRT-ACS in the experimental map.

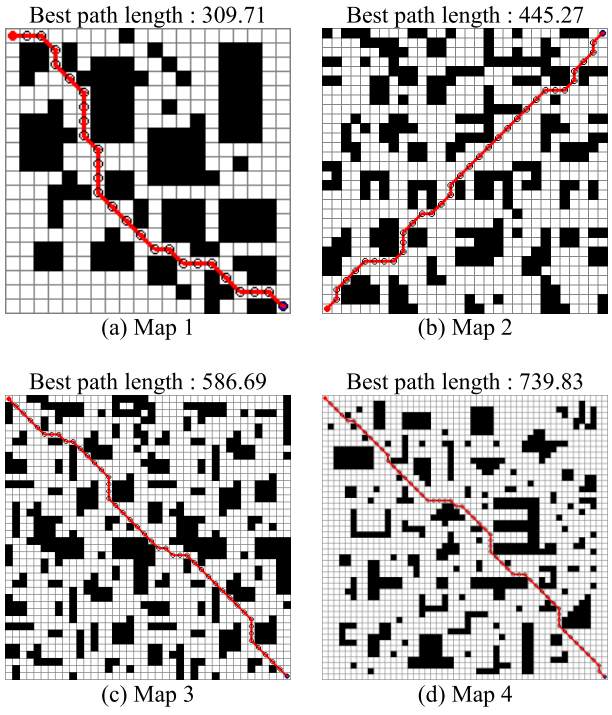


FIGURE 19. The best path generated by AIACSE in the experimental map.

measurements include the best path length, the worst path length, the mean path length, and the standard deviation.

Based on Table 9, Fig. 18 and Fig. 19, the advantages of the RRT-ACS algorithm can be seen over several ACO

TABLE 9. Statistical results obtained by each algorithm in different maps. The best result for each section is highlighted in bold.

Environments	Algorithm	Best	Worst	Mean	Std
Map 1	ACO	315.56	335.56	326.76	4.941
	RAS	309.71	321.42	311.66	4.441
	ACS	309.71	309.71	309.71	**0**
	PS-ACO	315.56	374.56	333.36	15.10
	AIACSE	309.71	309.71	309.71	**0**
	RRT-ACS	**290.62**	**302.04**	**294.18**	2.74
Map 2	ACO	508.70	591.13	559.65	19.14
	RAS	445.27	496.98	461.77	11.41
	ACS	445.27	453.55	445.74	1.28
	PS-ACO	453.55	524.26	475.88	17.94
	AIACSE	445.27	445.27	445.27	**0**
	RRT-ACS	**426.03**	**427.12**	**426.87**	3.31
Map 3	ACO	768.33	883.26	831.78	32.61
	RAS	586.69	664.26	627.63	24.82
	ACS	586.69	606.69	590.89	5.73
	PS-ACO	598.41	698.41	651.33	23.96
	AIACSE	586.69	594.98	**587.24**	**2.10**
	RRT-ACS	**581.51**	**590.54**	588.02	2.56
Map 4	ACO	1103.68	1289.53	1215.34	50.38
	RAS	739.83	888.11	812.51	34.09
	ACS	739.83	776.40	760.24	9.27
	PS-ACO	764.68	912.96	820.11	32.22
	AIACSE	739.83	748.11	743.96	**3.78**
	RRT-ACS	**712.54**	**723.47**	**717.77**	4.11

algorithms. However, it should be noted that the ACO algorithm uses discrete techniques (grid map) for path-planning. The RRT-ACS algorithm does not require this discretization and can be applied to continuous maps.

J. THE COMPARISON OF THE RRT-ACS ALGORITHM WITH OTHER RRT AND ACO HYBRID METHOD

To the best of the authors' knowledge, another researcher who uses the RRT and ACO hybrid method for path planning problems is Viseras *et al.* [36]. Viseras proposed a path planning algorithm with ants for continuous domains using RRT* and $ACO_{\mathbb{R}}$. Viseras named it as ACO-RRT*.

The RRT-ACS algorithm shows better performance compared with ACO-RRT*. Viseras tested the ACO-RRT* algorithm in the case of multiple rectangles, narrow passages, and mazes. Viseras reported that the paths generated by the ACO-RRT* algorithm were 0.6%-5.3% better than those generated by the RRT* algorithms. In this paper, the RRT-ACS algorithm is also tested in cases similar to the case used by Viseras, namely clutter (benchmark III), narrow passages (benchmark VIII), and maze (benchmark IX) cases. The result is that the paths generated by the RRT-ACS algorithm are 2.9%-9.5% better than those generated by the RRT* algorithm.

V. CONCLUSION

In this paper, the proposed path planning algorithm uses a hybridization of the RRT and ACS algorithms. Several comparative tests are carried out to evaluate the performance of the proposed algorithm. Statistical tests have also been carried out to verify whether there is a significant difference in performance between the proposed algorithm and other existing algorithms.

According to comparative tests between RRT-ACS, informed RRT*, informed RRT*-connect, RRT*, and RRT*-connect, the RRT-ACS algorithm has a good convergence speed. The results from benchmark case tests with known optimal values indicate that the proposed algorithm succeeded in reaching those optimal values. Furthermore, the statistical results show that the RRT-ACS algorithm has good performance compared to those of other existing algorithms with respect to the best path length, worst path length, mean path length, and standard deviation of the path length. We also discuss the stability, robustness, convergence, and rapidity of the RRT-ACS algorithm.

Based on the test results regarding the effect of exploitation and the exploration parameter (q_0), the contribution of adding the ACS algorithm to the proposed algorithm can be seen. When the contribution of the ACS algorithm is minimal (q_0 is between 0.1 and 0.3), the algorithm acts like the RRT and takes longer to obtain the optimal path. However, when q_0 is close to 0.5, the contribution of the ACS algorithm can be seen in speeding up the algorithm to build the optimal path.

REFERENCES

[1] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016, doi: 10.1109/TIV.2016.2578706.

[2] U. Lee, S. Yoon, H. Shim, P. Vasseur, and C. Demonceaux, "Local path planning in a complex environment for self-driving car," in *Proc. 4th Annu. IEEE Int. Conf. Cyber Technol. Autom., Control Intell.*, Hong Kong, Jun. 2014, pp. 445–450, doi: 10.1109/CYBER.2014.6917505.

[3] W. Sun and R. Alterovitz, "Motion planning under uncertainty for medical needle steering using optimization in belief space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, IL, USA, Sep. 2014, pp. 1775–1781, doi: 10.1109/IROS.2014.6942795.

[4] Y. Xiangwei and H. Liu, "A study on path planning of crowd animation based on PSO," in *Proc. 6th Int. Conf. Natural Comput.*, Yantai, China, Aug. 2010, pp. 2566–2570, doi: 10.1109/ICNC.2010.5583206.

[5] T. Ju, S. Liu, J. Yang, and D. Sun, "Apply RRT-based path planning to robotic manipulation of biological cells with optical tweezer," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, Beijing, China, Aug. 2011, pp. 221–226, doi: 10.1109/ICMA.2011.5985660.

[6] R. C. Jackson and M. C. Cavusoglu, "Needle path planning for autonomous robotic surgical suturing," in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, May 2013, pp. 1669–1675, doi: 10.1109/ICRA.2013.6630794.

[7] M. F. Abdelwahed, A. E. Mohamed, and M. A. Saleh, "Improving solution quality for experience-based framework through clustering algorithms," *IEEE Access*, vol. 7, pp. 106720–106725, 2019, doi: 10.1109/ACCESS.2019.2932893.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Seattle, WA, USA, May 2015, pp. 3067–3074, doi: 10.1109/ICRA.2015.7139620.

[9] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 6, pp. 1–54, 2017, doi: 10.3390/machines5010006.

[10] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Chicago, IL, USA, Sep. 2014, pp. 2997–3004, doi: 10.1109/IROS.2014.6942976.

[11] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[12] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 1478–1483, doi: 10.1109/ICRA.2011.5980479.

[13] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.

[14] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[15] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Atlanta, GA, USA, Dec. 2010, pp. 7681–7687, doi: 10.1109/CDC.2010.5717430.

[16] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, USA, Sep. 2011, pp. 2640–2645, doi: 10.1109/IROS.2011.6095077.

[17] L. Yuncheng and S. Jie, "A revised Gaussian distribution sampling scheme based on RRT* algorithms in robot motion planning," in *Proc. 3rd Int. Conf. Control, Autom. Robot. (ICCAR)*, Nagoya, Japan, Apr. 2017, pp. 22–26, doi: 10.1109/ICCAR.2017.7942654.

[18] W. Wang, H. Gao, Q. Yi, K. Zheng, and T. Gu, "An improved RRT* path planning algorithm for service robot," in *Proc. IEEE 4th Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Chongqing, China, Jun. 2020, pp. 1824–1828, doi: 10.1109/ITNEC48623.2020.9085226.

[19] A. H. Qureshi, S. Mumtaz, K. F. Iqbal, Y. Ayaz, M. S. Muhammad, O. Hasan, W. Y. Kim, and M. Ra, "Triangular geometry based optimal motion planning using RRT*-motion planner," in *Proc. IEEE 13rd Int. Workshop Adv. Motion Control (AMC)*, Yokohama, Japan, Mar. 2014, pp. 380–385, doi: 10.1109/AMC.2014.6823312.

[20] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia (ICRA)*, San Francisco, CA, USA, vol. 2, Apr. 2000, pp. 995–1001, doi: 10.1109/ROBOT.2000.844730.

[21] J.-G. Kang, D.-W. Lim, Y.-S. Choi, W.-J. Jang, and J.-W. Jung, "Improved RRT-connect algorithm based on triangular inequality for robot path planning," *Sensors*, vol. 21, no. 2, p. 333, Jan. 2021.

[22] S. Klemm, J. Oberlander, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "RRT*-connect: Faster, asymptotically optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Zhuhai, China, Dec. 2015, pp. 1670–1677, doi: 10.1109/ROBIO.2015.7419012.

[23] J. Wang, C. X.-T. Li, W. Chi, and M. Q.-H. Meng, "Tropistic RRT*: An efficient planning algorithm via adaptive restricted sampling space," in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, Aug. 2018, pp. 1639–1646, doi: 10.1109/ICInfA.2018.8812520.

[24] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy, 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.

[25] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[26] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 622–627.

[27] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[28] A. Yang, L. Gao, and Y. Luo, "An improved ant colony system algorithm for optimal path planning problem of mobile robots," in *Proc. 2nd Int. Conf. Comput. Modeling Simulation*, Sanya, Hainan, Jan. 2010, pp. 526–530, doi: 10.1109/ICCMS.2010.190.

[29] S. K. Calik, "UAV path planning with multiagent ant colony system approach," in *Proc. 24th Signal Process. Commun. Appl. Conf. (SIU)*, Zonguldak, Turkey, May 2016, pp. 1409–1412, doi: 10.1109/SIU.2016.7496013.

[30] S. Zhang, J. Pu, and Y. Si, "An adaptive improved ant colony system based on population information entropy for path planning of mobile robot," *IEEE Access*, vol. 9, pp. 24933–24945, 2021, doi: 10.1109/ACCESS.2021.3056651.

[31] Q. Song, Q. Zhao, S. Wang, Q. Liu, and X. Chen, "Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization," *IEEE Access*, vol. 8, pp. 62107–62115, 2020, doi: 10.1109/ACCESS.2020.2984695.

[32] X. Wang, H. Shi, and C. Zhang, "Path planning for intelligent parking system based on improved ant colony optimization," *IEEE Access*, vol. 8, pp. 65267–65273, 2020, doi: 10.1109/ACCESS.2020.2984802.

[33] C. Zhang, C. Hu, J. Feng, Z. Liu, Y. Zhou, and Z. Zhang, "A self-heuristic ant-based method for path planning of unmanned aerial vehicle in complex 3-D space with dense U-type obstacles," *IEEE Access*, vol. 7, pp. 150775–150791, 2019, doi: 10.1109/ACCESS.2019.2946448.

[34] Y.-T. Hsiao, C.-L. Chuang, and C.-C. Chien, "Ant colony optimization for best path planning," in *Proc. IEEE Int. Symp. Commun. Inf. Technol. (ISCIT)*, vol. 1, Oct. 2004, pp. 109–113.

[35] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014, doi: 10.1109/ACCESS.2014.2302442.

[36] A. Viseras, R. O. Losada, and L. Merino, "Planning with ants: Efficient path planning with rapidly exploring random trees and ant colony optimization," *Int. J. Adv. Robot. Syst.*, vol. 13, no. 5, pp. 1–16, 2016, doi: 10.1177/1729881416664078.

[37] P. Guo and L. Zhu, "Ant colony optimization for continuous domains," in *Proc. 8th Int. Conf. Natural Comput.*, May 2012, pp. 758–762, doi: 10.1109/ICNC.2012.6234538.

[38] B. Raveh, A. Enosh, and D. Halperin, "A little more, a lot better: Improving path quality by a path-merging algorithm," *IEEE Trans. Robot.*, vol. 27, no. 2, pp. 365–371, Apr. 2011, doi: 10.1109/TRO.2010.2098622.

[39] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. Congr. Evol. Comput. (CEC)*, vol. 2, Jul. 1999, pp. 1470–1477, doi: 10.1109/CEC.1999.782657.

[40] R. Jangra and R. Kait, "Analysis and comparison among ant system; ant colony system and max-min ant system with different parameters setting," in *Proc. 3rd Int. Conf. Comput. Intell. Commun. Technol. (CICT)*, Feb. 2017, pp. 1–4, doi: 10.1109/CIACT.2017.7977376.

[41] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3706–3712, doi: 10.1109/ICRA.2011.5980286.

[42] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT*," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 7, pp. 1–12, 2013, doi: 10.5772/56718.

[43] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006, doi: 10.1109/MCI.2006.329691.

[44] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004.

[45] B. Bullnheimer, R. F. Hartl, and C. Strauss, "A new rank based version of the ant system. A computational study," *Central Eur. J. Oper. Res. Econ.*, vol. 7, no. 1, pp. 25–38, 1999.

[46] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "RRT*-smart: Rapid convergence implementation of RRT* towards optimal solution," in *Proc. IEEE Int. Conf. Mechatronics Automat.*, Aug. 2012, pp. 1651–1656, doi: 10.1109/ICMA.2012.6284384.

[47] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011, doi: 10.1016/j.swevo.2011.02.002.

[48] M.-C. Kim and J.-B. Song, "Informed RRT* towards optimality by reducing size of hyperellipsoid," in *Proc. IEEE Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2015, pp. 244–248, doi: 10.1109/AIM.2015.7222539.

[49] M. Mavrovouniotis, G. Ellinas, and M. Polycarpou, "Ant colony optimization for the electric vehicle routing problem," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 1234–1241, doi: 10.1109/SSCI.2018.8628831.

[50] Y. Xue, X. Zhang, S. Jia, Y. Sun, and C. Diao, "Hybrid bidirectional rapidly-exploring random trees algorithm with heuristic target graviton," in *Proc. Chin. Autom. Congr. (CAC)*, Oct. 2017, pp. 4357–4361, doi: 10.1109/CAC.2017.8243546.

[51] C. Luan and Z. Fang, "Random particles boosted RRT for complicated 3D environments with narrow passages," in *Proc. 12nd World Congr. Intell. Control Autom. (WCICA)*, Jun. 2016, pp. 3271–3276, doi: 10.1109/WCICA.2016.7578529.

[52] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[53] A. Wu, S. Sadraddini, and R. Tedrake, "R3T: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 4245–4251, doi: 10.1109/ICRA40945.2020.9196802.
[54] G. Dong, W. W. Guo, and K. Tickle, "Solving the traveling salesman problem using cooperative genetic ant systems," *Expert Syst. Appl.*, vol. 39, no. 5, pp. 5006–5011, Apr. 2012, doi: 10.1016/j.eswa.2011.10.012.

MUHAMMAD ARIA RAJASA POHAN received the bachelor's degree in engineering physics and the master's degree in electrical engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, where he is currently pursuing the Doctoral degree in electrical engineering.

His research interests include robotics and artificial intelligence.

BAMBANG RIYANTO TRILAKSONO (Member, IEEE) received the degree in electrical engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, and the master's and Ph.D. degrees in electrical engineering from Waseda University, Tokyo, Japan.

He is currently a Professor with the Control and Computer System Research Group, School of Electrical Engineering and Informatics, ITB. His research interests include control systems, robotics, and artificial intelligence.

SIGIT PUJI SANTOSA received the degree (Hons.) in mechanical engineering/structural mechanics from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 1991, and the master's and Sc.D. degrees in mechanical engineering from the Massachusetts Institute of Technology (MIT), USA.

He is currently the Chairperson of the National Center for Sustainable Transportation Technology (NCSTT), ITB. His research interests include hybrid and electric vehicles, extended-range electric vehicles (EREVs), solid mechanics and plasticity, computational structural mechanics, CAD/CAE, structural crashworthiness/blastworthiness, occupant protection, ultralight metal body structures, armored fighting vehicles, and product development, such as car, SUV, bus, and LRT.

ARIEF SYAICHU ROHMAN (Member, IEEE) was born in Malang, East Java, Indonesia. He received the bachelor's (Ir.) degree in electrical engineering from the Institut Teknologi Bandung (ITB), Indonesia, in 1990, the M.Eng.Sc. degree in systems and control from the University of New South Wales, Sydney, Australia, in 1998, and the Ph.D. degree in systems and control from The University of Newcastle, Newcastle, Australia, in 2005.

He was with the Research and Development Division, PT IPTN, the Indonesian aircraft industry, from 1990 to 1992. He has been teaching undergraduate and graduate courses in electrical engineering with the School of Electrical Engineering and Informatics, ITB, since 1992, where he served as the Head for the undergraduate program in electrical engineering, from 2012 to 2016. He has published papers in various national and international journals and conferences. His research interests include anti-windup systems, model predictive control, sliding-mode control and its applications (distillation columns), electric vehicles, and autodrive systems.