electronics

MDPI

Article

A Performance Review of Collision-Free Path Planning Algorithms

Hyunwoo Shin and Junjae Chae *

School of Air Transport, Transportation and Logistics, Korea Aerospace University, 76 Hanggongdaehak-ro, Deogyang-gu, Goyang-si, Gyeonggi-do 10540, Korea; hyunwoo4171@kau.kr

* Correspondence: jchae@kau.ac.kr

Received: 12 January 2020; Accepted: 8 February 2020; Published: 12 February 2020

Abstract: Path planning for mobile agents is one of the areas that has drawn the attention of researchers', as evidenced in the large number of papers related to the collision-free path planning (CFPP) algorithm. The purpose of this paper is to review the findings of those CFPP papers and the methodologies used to generate possible solutions for CFPP for mobile agents. This survey shows that the previous CFPP papers can be divided based on four characteristics. The performance of each method primarily used to solve CFPP in previous research is evaluated and compared. Several methods are implemented and tested in same computing environment to compare the performance of generating solution in specified spatial environment with different obstacles or size. The strengths and weakness of each methodology for CFPP are shown through this survey. Ideally, this paper will provide reference for new future research.

Keywords: collision-free; path planning; robot motion planning; mover's problem; heuristic based algorithms

1. Introduction

The usage of mobile agents (MAs) including unmanned aerial vehicles (UAVs), autonomous robots and automated guided vehicles (AGVs), in areas, such as container terminals, delivery to customers or manufacturing processes only continues to increase [1]. The design problem of Collision-free path planning (CFPP) is to determine a collision-free path of MAs within the target space of a specific environment [2]. Research also refers to this design problem as motion planning or piano mover's problem [3]. The researchers have studied CFPP problem while using several kinematic constraints, including point robots or using robots with specific degree of freedom (DOF) [4–6]. Autonomous robots are also required to decide their motion using the logic. Researches have conducted extensive research in response to the increasing importance of MAs.

In this paper, we have arranged the survey papers of the CFPP based problem according to the paper's original research purpose. Schwartz & Sharir [7] focused on classical geometry, topology, algebraic geometry, algebra, and combinatorics. Additionally, they briefly discuss the characteristic and complexity of geometric algorithms that are relevant to the CFPP problem. Hwang & Ahuja [8] researched the different methodologies, obstacles, and algorithms, and discussed the characteristics of the problem known as the gross motion plan. Sariff & Buniyamin [9] reviewed the algorithms for solving CFPP problem in different environments. The study further briefly presented the advantages and disadvantages of the algorithms. Masehian and Sedighizadeh [10] conducted a 35 years period chronological survey regarding the classic and meta-heuristic algorithms for the CFPP problem. Goerzen et al. [11] found that UAVs have significant differing issues than that of other robots, because it operates in a three-dimensional (3D) environment. Roberge et al. [12] reviewed the performance of algorithms, parallel genetic algorithm, and particle swarm optimization pertaining to the non-deterministic CFPP problem. In addition, the researchers examined the dynamic

properties of a UAV's path in a complex three-dimensional environment. A survey of heuristic based algorithm for solving CFPP was recently presented [13]. The research summarized chronological changes and sorted studies by algorithms, which were nature-inspired methods, fuzzy logic, neural networks, and the potential field method. The frequency of the problem attributes is also reviewed in the paper. Schwarting et al. [14] reviewed and provided emerging trends in autonomous problems. The paper summarized problems, such as integration of automation and safety guarantees when applied to robots.

Researchers in different fields have conducted similar studies; hence, there are problems with regards to the definitions and terms. For example, some of the papers use the name of problems as mover's problem [15], motion planning problem [16] or path planning problem [2]. Numerous methodologies are provided in the research while using different key words [5,13,17,18]. However, a comparison of methodologies is rarely studied. Previous research has focused on the different characteristics of the CFPP problem, such as sampling-based algorithm [19], or has used a very small number of searching algorithms [12]. On the other hand, researches have not focused on the differences in terminology and meanings. In the writers' opinion, this lack of uniformity and conformity in terminology makes it difficult to select and decide on an appropriate algorithm. Therefore, the objectives of this study are as follows. First, the purpose of this research is to unify the name and attributes. Secondly, to provide a method to classify algorithms for solving the problem. Third, the performance of the algorithms for solving CFPP is measured within a fundamental environment. These contributions will enable researchers, who study the CFPP problem to grasp the key point in a problem. Finally, researchers will be better able to choose searching algorithms, through the comparison of the different algorithms.

The rest of this paper is divided for four sections. The CFPP problem is defined in Section 2 and four different categories of CFPP problem are proposed. In Section 3, a plan of experiment for searching algorithms performance review is briefly introduced. Flowcharts of the algorithms are shown and environments for testing are suggested. Additionally, a method for tuning parameters, which are components of metaheuristic algorithms, is concisely shown. The results of the experiments are organized and analyzed in Section 4. In the final section, a conclusion and a summary are presented.

2. CFPP

The goal of collision-free path planning (CFPP) problem is helping MAs or robots to find a safe path from the starting point to ending (goal) point [2]. The path from a result of the CFPP problem is considered to be an effective bypass for MAs. The effectiveness is discussed from this perspective, which is the shortest distance for MAs. Additionally, this can be researched with multi-objective model, such as reducing resource consumption with shorter distances or finding a good solution in shorter time. Four different attributes are required to define the problem type; environment type (Env.T.; Section 2.1), environmental representation (Env.R.; Section 2.2), searching algorithm (S.Alg.; Section 2.3), and experimental type (Exp.T.; Section 2.4). Additionally, related works that emphasize and improve at least one point of any one of these characteristics are necessary to define the problem type. Details of the categories are shown in Figure 1.

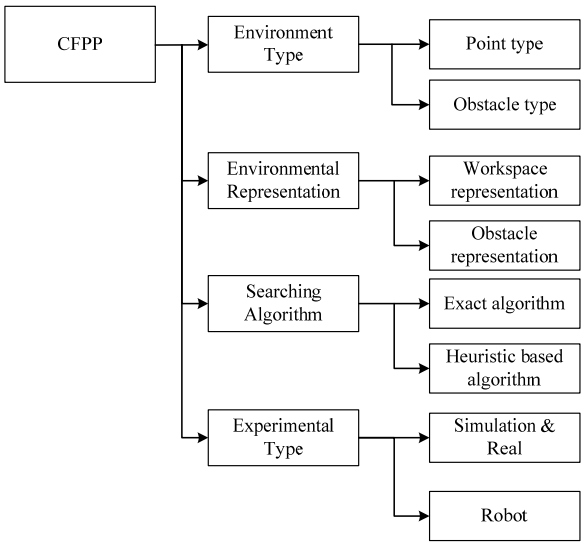


Figure 1. Characteristics of the collision-free path planning (CFPP).

In order to deliver information efficiently, this paper uses shorthand notation. The notation is organized in Table 1.

Table 1. Shorthand notation for the tables.

Word	Notation	Word	Notation
Environment type	Env.T.	Network	N.
Environment representation	Env.R.	Coordinate system	Co.S.
Searching algorithm	S.Alg.	Boundary representation	B.R.
Experimental Type	Exp.T.	Cell tree	C.T.
Obstacle type	O.T.	Polygonal approximation	Po.A.
Point type	P.T.	Evolutionary programming	E.P.
Workspace representation	W.R.	K nearest neighbor	K near.
Obstacle representation	Ob.R.	Simulation	S.
Experiment	Exp.	Probabilistic roadmap	PRM
Workspace representation algorithm	W.R.Alg.	Rapidly-exploring Random Tree	RRT
Certain	C.	Voronoi Diagram	V.D.
Uncertain	Uc.	Visibility Graph	Vgraph
Static	S.	Normal distribution transform	NDT
Dynamic	D.	Circle approximation	Ci.A

2.1. Environment Type

Environment types in CFPP problem are divided specifically, as follows: obstacle type (O.T.) and point type (P.T.). Each type has specific characteristics and detail descriptions of the characteristics are proposed as follows.

Firstly, the obstacle type is classified as static or dynamic [6,20]. Figure 2a describes static obstacle type and Figure 2b is an example of dynamic obstacle type. The static obstacle environment, which is also called the stationary obstacle environment, is a basic form of the CFPP problem. In this case, the obstacles do not change their shape or position. The CFPP problem in a static environment is the most basic model of the CFPP. This is one of the reasons why many papers use a static environment assumption [13]. The dynamic obstacle environment, which is also called the moving obstacle, can change the obstacle's shape time dependently. This case considers the velocity and acceleration of MAs.

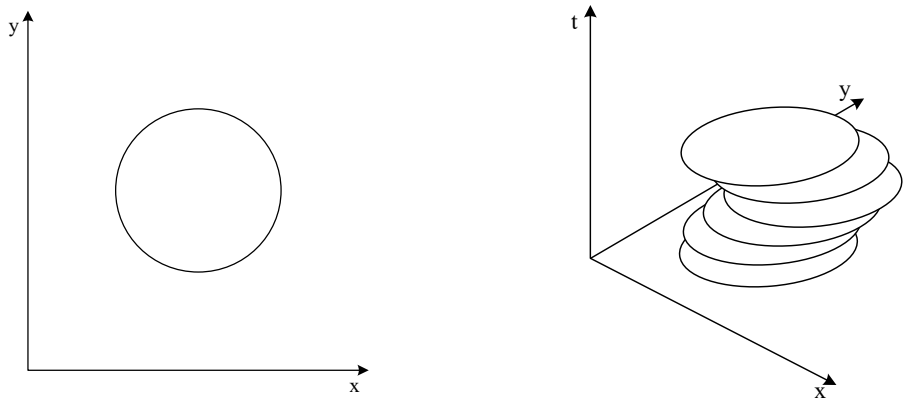


Figure 2. Illustrations of the obstacle having different obstacle type: (**a**) Static obstacle; and, (**b**) Dynamic obstacle.

Second, point type is classified as certain or uncertain [21]. As there is inherent uncertainty in the MA's sensing, point type should be assumed as certain or uncertain. Certain point type is assumed when the MAs position and goal point is an exact location. The assumption is often used in testing novel approaches while using simulation, because uncertain problem makes it difficult to solve the problem. Uncertain point type is set when the MA's point and goal point is not known exactly. MAs use several sensors to recognize space. The sensors may include some errors, which MAs must recognize in an uncertain space. The type makes the problem more challenging than with a certain point type. This is one of the reasons why it is difficult to solve a CFPP problem with real MAs. The problem is solved while using the stochastic technique, such as the Monte-Carlo method [22–24].

2.2. Environmental Representation

Environmental representations in a CFPP problem have two different types: workspace representation (W.R.) and obstacle representation (Ob.R.). Diverse studies about these have been previously researched. The details of these studies are characterized below.

First, workspace representation has two representations: the coordinate system model and the network model. The coordinate system model uses Algebra for describing the environment and the MAs' motion. This problem is called the Findspace problem, which is one aspect of spatial planning [25]. The Findspace problem has three different workspaces; the polygonal workspace; the polyhedral workspace; and, the semi-Algebraic workspace [21]. In addition, convexity and non-convexity are also important aspects of the Findspace problem. Simply, the greater the number of edges and vertices in a workspace representation, the greater the complexity of the problem. Therefore, a polyhedral workspace would be more difficult to solve for than a polygonal workspace [6]. Network model, which is also known as the Roadmap approach, uses links and nodes for describing the environment. In this model, researchers have modelled the routes of the MA's as links and nodes using algorithms, such as the Voronoi diagram, the Visibility Graph, the Rapidly-exploring random tree, and the Probabilistic roadmap. Additionally, the algorithms are mainly differentiated from each other mainly by style generated. These include: the sampling-based algorithm and the geometric modeling algorithm [17]. LaValle et al. [26] conducted extensive researched of the sampling-based algorithm while using the Monte Carlo sampling method.

Their conclusions suggest that the basic grid network algorithm provides better performance in certain cases than does a sampling-based algorithm. Thrun et al. [23] proposed a model wherein the Monte Carlo localization is combined with the sampling-based algorithm. Janson et al. [24] posit that the Monte Carlo Motion Planning (MCMP) is effective in solving the CFPP problem. Englot and

Hover [27] suggested the sampling-based algorithm under ACO framework to solve multi-goal CFPP problems. This technique allows for a solution that can take into account multi-objects, such as computational complexity and optimality. Ichter et al. [28] studied the sampling-based algorithm in a learning-based framework where non-uniform sampling strategies are used for the CFPP problem in "huge" spaces.

The sampling-based algorithm includes two common algorithms: the Probabilistic roadmap and the Rapidly-exploring random tree. Kavraki et al. [29] solve the CFPP problem for articulated robots while using two phases: the Probabilistic roadmap and the heuristic for searching within the roadmap. The proposed method can be applied to any holonomic robot. The theoretical performance of the Probabilistic roadmap was rarely studied before Kavraki et al. [19]. The paper studies the failure probability of connecting two points. Also, the relationship among the number of nodes, failure probability, and length of paths are studied. The Probabilistic roadmap and the Rapidly-exploring random tree are also studied in practical way with various robots [30,31]. Several simulated and real experiments using specific robots are performed. Complex and large environments, such as dynamic obstacles or urban areas, are considered in the experiments. Kim et al. [32] propose the Tangent Bundle Rapidly exploring random tree (TB-RRT) in order to improve the efficiency of the algorithm in a large space. Also, related works under complex environment are studied in succession [33,34]. A centralized case, such as spot-welding station with two to six robot manipulators in automobile manufacturing field, is researched and an application is made for the case [35,36]. These papers purport that centralized planning is more efficient than decentralized planning. Additionally, the sampling-based algorithms are applied to a geometrically complex problem and a multi-robot problem [37,38]. Wilmarth et al. [39] argue that the basic Probabilistic roadmap is an ineffective method for solving the CFPP problem in a narrow space. Thus, a novel algorithm, named MAPRM, is suggested for solving specific narrow space. Saha et al. [40] concluded the small-step retraction method can be applied to the CFPP problem. Meanwhile, the CFPP problem can be expanded to the tour problem, as can the traveling-salesman problem or task-planning problem [41,42]. The algorithms are mainly divided into two divisions: path finding and tour (sequence) finding in order to solve the problem,

Meanwhile, several papers have evaluated the performance of the sampling-based algorithm. The sampling configuration and probability measure are affected by the performance of the Probabilistic roadmap [43]. Further, Karaman and Frazzoli [44] studied the performance of the Probabilistic roadmaps and Rapidly-exploring random trees sampling-based algorithms. Recently, Marble and Bekris [45] discuss and introduce a method providing robust solution qualities. More details about related to sampling-based algorithm research are organized in Table 2.

Whatever path that is generated in a network model, should be smoothed before applying to MAs [17]. Smoothing affects the performance of the algorithm [46], and it has been studied by some papers [47,48].

Table 2. Classification of the papers while using the sampling-based algorithm.

Ref.	Author & year	Env.T.		Env.R		S.Alg.	Exp.T.		W.R.Alg.
		P.T.	O.T	W.R.	Ob.R		Exp.	Robot	
[29]	Kavraki et al., 1996	C.	S.	N.	Grid	Heuristic	S.	Articulated robots	PRM
[19]	Kavraki et al., 1998	C.	-	-	B.R.	-	S.	6 DOF	PRM
[49]	Hsu et al., 1997	C.	S.	N.	-	Heuristic	S.	6 DOF	PRM
[39]	Wilmarth et al., 1999	C.	S.	N.	B.R.	-	S.	-	MAPRM
[31]	Kuffner & LaValle, 2000	C.	S.	N.	-	Heuristic	S.	7 DOF	RRT
[23]	Thrun et al., 2001	Uc.	S.	N.	B.R., C.T.	MCL algorithm	S.	RWI B18 robot	Sampling based
[35]	Sánchez & Latombe, 2002	C.	D.	N.	Po.A.	Heuristic	S.	Multi robot	PRM

[36]	Sanchez & Latombe, 2002	Uc.	D.	N.	Po.A.	Heuristic	S.	-	PRM
[37]	Sánchez & Latombe, 2003	C.	D.	N.	-	Heuristic	S.	6 robots	PRM
[26]	LaValle et al., 2004	-	-	N.	-	-	S.	-	Sampling based review
[40]	Saha et al., 2005	C.	S.	N.	-	F*	S.	-	PRM
[43]	Hsu et al., 2006	Uc.	S.	N.	-	Heuristic	S.	-	PRM
[41]	Saha et al., 2006	-	S.	N.	-	Heuristic	S.	-	PRM
[50]	Alterovitz et al., 2007	Uc.	S.	N.	B.R.	Heuristic	S.	-	Sampling based
[30]	Hsu et al., 2002	Uc.	D.	N.	B.R.	Heuristic	Real	-	PRM
[34]	Kuwata et al., 2009	Uc.	D.	N., Co.S.	Grid	Heuristic	S.	Vehicle	RRT
[27]	Englot & Hover, 2011	C.	S.	N.	-	ACO	S.	-	Sampling based
[44]	Karaman & Frazzoli, 2011	C.	S.	N.	B.R.	Heuristic	S.	-	Sampling based
[33]	Malone et al., 2014	-	D.	N.	Grid	Dijkstra	S.	-	PRM
[24]	Janson et al., 2018	Uc.	S.	N.	B.R.	MCMP	S.	-	Sampling based
[51]	Contreras-Cruz et al., 2015	Uc.	S.	N.	-	EP, Dijkstra	Real	Xidoo-Bot	PRM
[42]	Dantam et al., 2016	-	S.	N.	-	Heuristic	Real	-	RRT
[38]	Solovey et al., 2016	-	S.	N.	-	K Near., M*	S.	High DOF Multi robot	RRT
[32]	Kim et al., 2016	-	S.	N.	-	K Near.	S.	-	RRT
[45]	Marble & Bekris, 2017	C.	S.	N.	-	K Near.	S.	-	PRM
[28]	Ichter et al., 2018	-	D.	N.	-	Heuristic	S.	Multi robots	Sampling based

Secondly, Hwang and Ahuja [8] propose six basic obstacle representations that are chosen in consideration of workspace representation and an approximation of the obstacles. These six obstacle representations are: original objects, gird, cell tree, polygonal approximation, constructive solid geometry, and boundary representation. The representation styles are shown in Figure 3. These obstacle representations are chosen by workspace representation and approximation of the obstacles.

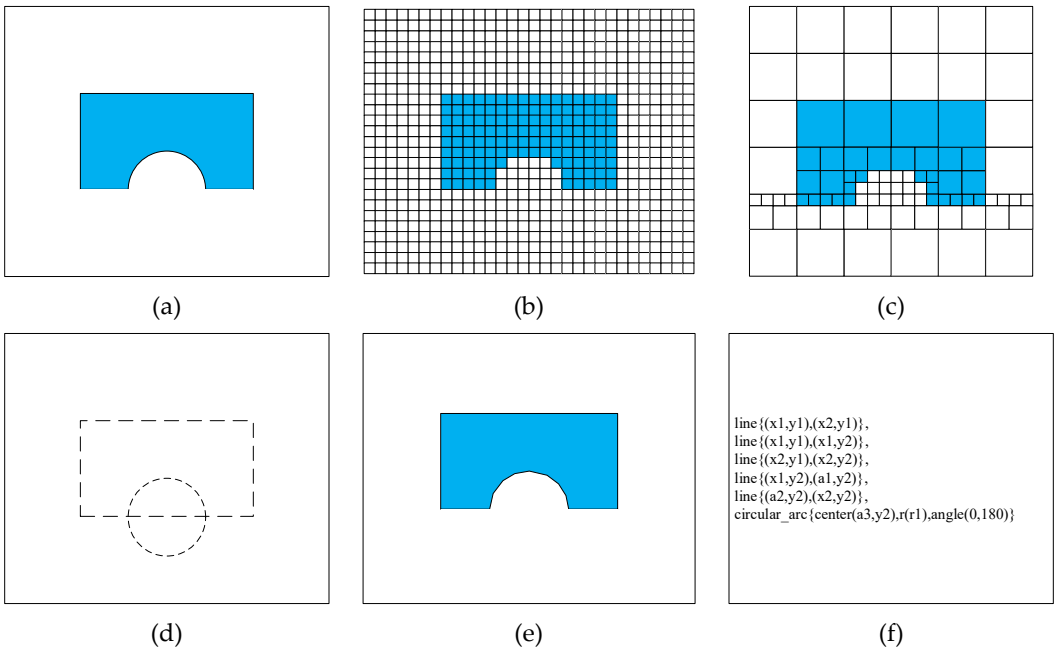


Figure 3. Six different workspace representations [8]: (**a**) Original object; (**b**) Grid; (**c**) Cell tree; (**d**) Constructive solid geometry; (**e**) Polygonal approximation; and, (**f**) Boundary representation.

2.3. Searching Algorithms

Searching algorithms or query process in probabilistic roadmaps are chosen by workspace representation. Some searching algorithms are applied to specific representations. For example, A* and Dijkstra algorithms are only applicable under network conditions. For instance, to solve problem with network, A*, Dijkstra algorithm or heuristic-based algorithms are often used. On the other hand, optimal approaches are used for problem under a coordinate system. There is extensive research that provides searching algorithms for several conditions in reasonable time. The algorithms can be classified as classic algorithms or heuristic-based algorithms [10,13,18].

2.3.1. Classical Algorithm

Dijkstra Algorithm is a common used algorithm for solving the network problem [52]. This algorithm offers an optimal path in a network, when all of the network link distances are positive. The A* algorithm is also frequently used in order to solve the shortest path problem on network [53]. Some methodologies, which are a combination of a Voronoi diagram and the Dijkstra algorithm, are provided by several researches [54,55]. Additionally, the Dijkstra algorithm was studied as query process of probabilistic roadmaps [33,56]. An application was suggested for solving the CFPP problem while using the Dijkstra algorithm [57]. A* algorithm is broadly used to solve the CFPP problem under network conditions. Several research papers related to a visibility graph were applied to A* algorithm to solve CFPP problem [2,20,58]. On the other hand, some papers provide methods applied under a sampling-based environment [59,60]. Many papers suggested novel algorithms, which were developed using Dijkstra's algorithm or A* algorithm [61–63]. Papers about A* and Dijkstra algorithms are shown in Table 3.

Table 3. Classification of the papers using the exact algorithm.

Ref.	Author & year	Env.T.		Env.R		S.Alg.	Exp.T.		W.R.Alg.
		P.T.	O.T.	W.R.	Ob.R		Exp	Robot	
[54]	Bhattacharya & Gavrilova, 2007	-	D.	N.	B.R.	Dijkstra	S.	-	V.D.

[55]	Ho & Liu, 2009	-	S.	N.	B.R.	Dijkstra	S.	Car-like robot	V.D.
[56]	Janson et al., 2018	-	-	N.	B.R.	Dijkstra	S.	-	PRM
[57]	Wang et al., 2011	C.	S.	N.	B.R.	Dijkstra	S.	-	Grid
[20]	Alexopoulos & Griffin, 1992	C.	S.	N.	B.R.	A*	S.	A mobile robot	Vgraph
[64]	Fu & Liu, 1990	C.	S.	N.	Po. A.	A*	S.	-	Vgraph
[58]	Herman, 1986	C.	S.	N.	C.T.	A*	S.	-	C.T.
[2]	Lozano-Pérez & Wesley, 1979	C.	S.	N.	B.R.	A*	S.	-	Vgraph
[60]	Berg et al., 2006	Uc.	D.	N.	B.R.	D*	S.	-	Roadmap
[59]	Bohlin & Kavraki, 2000	C.	S.	N.	B.R.	A*	S.	6 DOF robot	PRM
[62]	Deng et al., 2012	Uc.	D.	N.	-	Dijkstra	S.	-	-
[63]	Duchoň et al., 2014	-	-	N.	Grid	A*	S.	-	SLAM based grid
[61]	Noto & Sato, 2002	-	-	N.	-	Dijkstra	S.	-	Grid

2.3.2. Heuristic Based Algorithm

There are many heuristic-based methods for the CFPP problem. Wavefront algorithm is one of the heuristics that adopts the idea of a potential field and it was applied with various characteristics. For instance, the Wavefront algorithm is used for solving a problem with three-dimensional normal distributions transformations [65]. Several kinds of the metaheuristic are applied to the CFPP problem that have a network or coordinate system.

Evolutionary algorithms, such as the Genetic algorithm (GA), are usually studied under network environments [66–69] that are related to CFPP. Additionally, several algorithms are developed for under coordinate system conditions [70–73]. Some novel algorithms that are based on GA are provided from related works [74–78]. GA is one of the most applicable algorithms to solve the CFPP problem with multi-robot [79]. Table 4 briefly shows those papers which use Wavefront or GA.

Table 4. Classification of the paper using the heuristic-based algorithm (Wavefront and Genetic algorithm (GA)).

Ref.	Author & year	Env.T.		Env.R		S.Alg.	Exp.T.		W.R.Alg.
		P.T.	O.T.	W.R.	Ob.R.		Exp.	Robot	
[65]	Stoyanov et al., 2010	C.	-	N.	-	Wavefront	S.	-	NDT
[66]	AL-Taharwa et al., 2008	C.	S.	N.	Grid	GA	S.	-	Grid
[67]	Cai & Peng, 2002	C.	S.	N.	B.R.	GA	S.	Two robots	Obstacle edge
[68]	Yang & Yoo, 2018	-	D.	N.	Grid	GA, ACO	S.	UAV	Grid Layer
[69]	Hu & Yang, 2004	C.	D.	N.	Grid	GA	S.	-	Grid
[73]	MahmoudZadeh et al., 2018	Uc.	D.	Co.S.	B.R.	EA	S.	AUV	-
[71]	Elshamli et al., 2004	-	D.	N.	B.R.	GA	S.	-	-
[70]	Jiang et al., 2018	C.	S.	Co.S.	B.R.	GA	S.	-	Mechanical arm
[72]	Zhao et al., 1994	C.	S.	Co.S.	B.R.	GA	S.	Mobile Manipulator	-
[78]	Tu & Yang, 2004	C.	D.	N.	Grid	GA	S.	-	Grid
[75]	Lamini et al., 2018	C.	S.	N.	Grid	GA	S.	-	Grid
[74]	Lee et al., 2018	C.	S.	N.	Grid	GA	S.	-	Grid
[77]	Sedighi et al., 2004	C.	S.	N.	Grid	GA	S.	-	Grid

[76]	Tuncer & Yildirim, 2012	C.	D.	N.	Grid	GA	S.	-	Grid
[79]	Nazarahari et al., 2019	C.	S.	Co.S.	B.R.	GA, Wavefront	S.	Multi-robot	Potential field

Simulated Annealing (SA) is typically combined with the potential field method [80–84]. The other searching algorithms, such as general heuristic algorithm, population-based metaheuristic algorithm and fuzzy logic, and environmental representations, are also researched with SA [85–90]. SA based algorithms have also been applied in CFPP problems to solve vehicle congestion in smart cities. [91]. Papers that are based on SA, which are cited above, are organized in Table 5.

Table 5. Classification of the paper using the heuristic-based algorithm (Simulated Annealing (SA)).

Ref.	Author & year	Env.T.		Env.R		S.Alg.	Exp.T.		W.R.Alg.
		P.T.	O.T.	W.R.	Ob.R.		Exp.	Robot	
[80]	Janabi-Sharifi & Vinke, 1993	C.	S.	Co.S.	B.R.	SA	S.	Disk robot	Potential field
[82]	Park et al., 2001	-	S.	Co.S.	B.R.	SA	Both	Mobile robot	Potential field
[84]	Park & Lee, 2002	-	S.	Co.S.	B.R.	SA	Both	Mobile robot	Potential field
[81]	Zhu et al., 2006	C.	S.	Co.S.	B.R.	SA	S.	-	Potential field
[90]	Carriker et al., 1990	C.	S.	Co.S.	B.R.	SA	S.	Mobile Manipulator	-
[89]	Kroumov & Yu, 2009	C.	S.	Co.S.	B.R.	SA, NN	S.	-	Potential field
[86]	Martínez-Alfaro & Gómez-García, 1998	C.	S.	Co.S.	B.R.	SA, Fuzzy	S.	-	-
[85]	Miao & Tian, 2008	C.	D.	N.	B.R.	SA	S.	-	Obstacle edge
[88]	Miao & Tian, 2013	C.	D.	N.	B.R.	SA	S.	-	Obstacle edge
[87]	Tavares et al., 2011	C.	S.	Co.S.	B.R.	SA	S.	-	-
[91]	Amer et al., 2019	C.	D.	N.	-	SA	S.	Vehicles	Road

Particle swarm optimization (PSO) was originally defined in a continuous environment, so much CFPP research is studied under a coordinate system [46,92–100]. Chen and Li [99] use a distinguishable method for obstacle representation, which is circle approximation (Ci.A.). They assume the obstacles as being a circle. On the other hand, some research provide methods, which are PSO-based algorithms in a network condition [101–104]. Some research has focused on solving the multi-robots CFPP problem while using PSO [105,106]. In Table 6, additional information from previous research related to PSO is shown.

Table 6. Classification of the paper using the heuristic-based algorithm (PSO).

Ref.	Author & year	Env.T.		Env.R		Exp.T.		W.R.Alg.
		P.T.	O.T.	W.R.	Ob.R.	Exp.	Robot	
[99]	Chen & Li, 2006	C.	S.	Co.S.	Ci.A.	S.	Car-like robot	-
[93]	Foo et al., 2007	-	S.	Co.S.	B.R.	S.	UAV	-
[95]	Fu et al., 2011	-	S.	Co.S.	B.R.	S.	UAV	-
[96]	Gong et al., 2011	C.	S.	Co.S.	B.R.	S.	-	-
[97]	Saska et al., 2006	C.	S.	Co.S.	B.R.	S.	Robotic soccer	-
[92]	Song et al., 2019	C.	S.	Co.S.	Grid	S.	-	Grid
[46]	Tharwat et al., 2019	C.	S.	Co.S.	B.R.	S.	-	-
[94]	Zhang et al., 2013	Uc.	S.	Co.S.	B.R.	S.	-	-
[98]	Zhang et al., 2013	C.	S.	Co.S.	B.R.	S.	UAV	-
[104]	Kang et al., 2008	C.	S.	N.	B.R.	S.	-	Obstacle edge
[100]	Masehian & Sedighizadeh,	C.	S.	Co.S.	B.R.	S.	-	-

	2010							
[102]	Phung et al., 2017	C.	S.	N.	-	S.	UAV	Vision-based inspection
[103]	Shiltagh & Jalal, 2013	C.	S.	N.	Grid	S.	-	Grid
[101]	Wang et al., 2015	C.	D.	N.	Grid	S.	-	Grid
[105]	Alejo et al., 2013	-	-	Co.S.	-	S.	Multi-UAV	-
[106]	Thabit & Mohades, 2019	C.	S.	N.	Grid	S.	Multi-UAV	Grid

Ant colony optimization (ACO) is usually studied under varying network environments, such as grid network and Voronoi diagram [107–119]. Few papers, likewise, researched GA with ACO and coordinate system [120]. A paper, which solves the CFPP problem with automated underwater vehicles (AUV), is studied [121]. This paper suggests an ACO based algorithm to solve large-scale CFPP problems. Table 7 organizes details regarding papers based on ACO.

Table 7. Classification of the paper using the heuristic-based algorithm (ant colony optimization (ACO)).

Ref.	Author & year	Env.T.		Env.R		Exp.T.		W.R.Alg.
		P.T.	O.T.	W.R.	Ob.R.	Exp.	Robot	
[112]	Akka & Khaber, 2018	C.	S.	N.	Gird	S.	-	Grid
[115]	Brand et al., 2010	C.	S.	N.	Grid	S.	-	Grid
[114]	Chia et al., 2010	C.	S.	N.	Grid	S.	-	Grid
[109]	Garcia et al., 2009	C.	D.	N.	Grid	S.	-	Grid
[108]	Jiao et al., 2018	C.	S.	N.	Grid	S.	Wheelchairs	Grid
[107]	Xing et al., 2011	C.	D.	N.	Grid	S.	-	Grid
[118]	Xiong et al., 2019	C.	S.	N.	Grid	Both	AMV	V.D.
[117]	Cong & Ponnambalam, 2009	C.	S.	N.	Grid	S.	-	Grid
[110]	Yen & Cheng, 2018	C.	S.	N.	Grid	S.	(Multi-task)	Grid
[111]	Hsiao et al., 2004	-	-	N.	-	S.	-	Random Generated
[113]	Yu et al., 2019	C.	S.	N.	B.R.	S.	AUV (Multi-task)	Cube, Dense
[116]	Zhang et al., 2010	C.	S.	N.	Grid	S.	UAV	Point
[120]	Fan et al., 2003	C.	S.	Co.S.	B.R.	S.	-	-
[119]	Wang et al., 2019	C.	S.	N.	Grid	S.	Ground robot	Cube
[121]	Ma et al., 2019	C.	S.	N.	Grid	S.	AUV	Cube

2.4. Experimental Type

In experimental type, two characteristics, which are the robot type and experimental method, should be decided. Many previous works researched under different types of the robot, such as car-like robots, UAV, humanoid, and several degree of freedom (DOF) robots. Each robot has its own moving limitation. This constraint is a cause of different kinematic constraints [21]. Kinematic constraints should be considered under experimental method. When making experiments with real robot, computation, and resources for solving the problem is restricted. On the other hand, the simulation method might have some assumptions, which are different from real robot. The assumptions make a problem simplified to solve and this simple problem contributes to test novel methodologies.

3. Experiments Plan

In this paper, a simulation experiment for performances is considered under simple environments and a single point robot, which is able to move without kinematic constraints. The experiment is contributed to measure the performance of basic searching algorithms, as follows; Dijkstra algorithm, A* algorithm, Wavefront algorithm, Genetic algorithm, Simulated annealing, Ant colony optimization, and particle swarm optimization. Environment type is assumed as certain point type and static obstacle. The experiment uses grid network in order to make environment simple, and obstacles are represented as grid.

The object of the algorithm is to find a shortest path. Therefore, the objective function of the comparison experiment is formulated, as follows [74].

$$min\, f(P_s, P_g) = d(P_s, P_1) + \sum_{i=1}^{n-1} d(P_i, P_{i+1}) + d(P_n, P_g), \quad (1)$$

The function is representing a path length from P_s to P_g on network and the meaning of $d(a, b)$ is network distance from node a to node b.

3.1. Algorithms

The Dijkstra algorithm, A* algorithm and Wavefront algorithm have been the most commonly applied algorithms applied to solve the CFPP problem. However, other metaheuristic algorithms have been applied, each a modified version based on flow and heuristics. The paper reviewed each simple form of each algorithm. The flows for each metaheuristic algorithm are shown in Figure 4, Figure 5, Figure 6 and Figure 7.

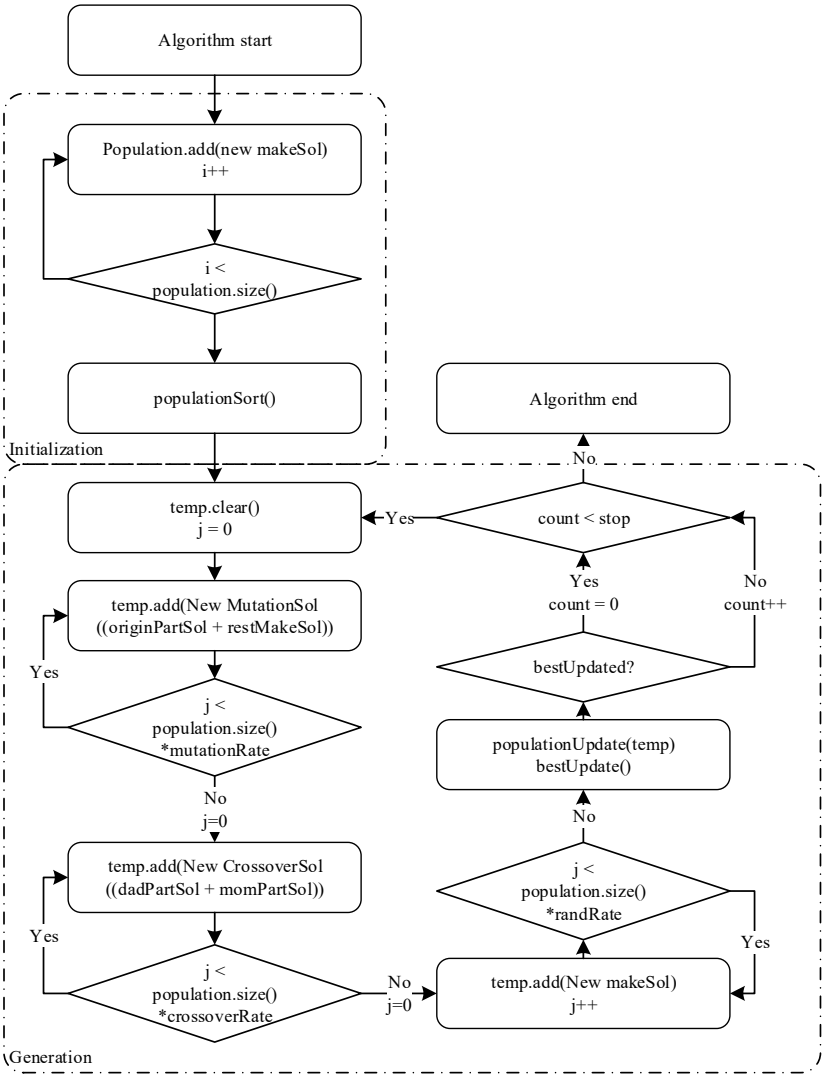


Figure 4. The flowchart of GA.

In Figure 4, flow of GA is divided into two parts: initialization and generation. The initialization part is initializing the GA's population set, which is then improved in Generation part. The generation part has three methods: crossover, mutation, and random generation. Each new population set is generated through these methods and the next generations are updated based on

the population set. The generation number is counted when the best solution is not updated and resets to zero when the best solution is updated. Executing a generation count completes the algorithm.

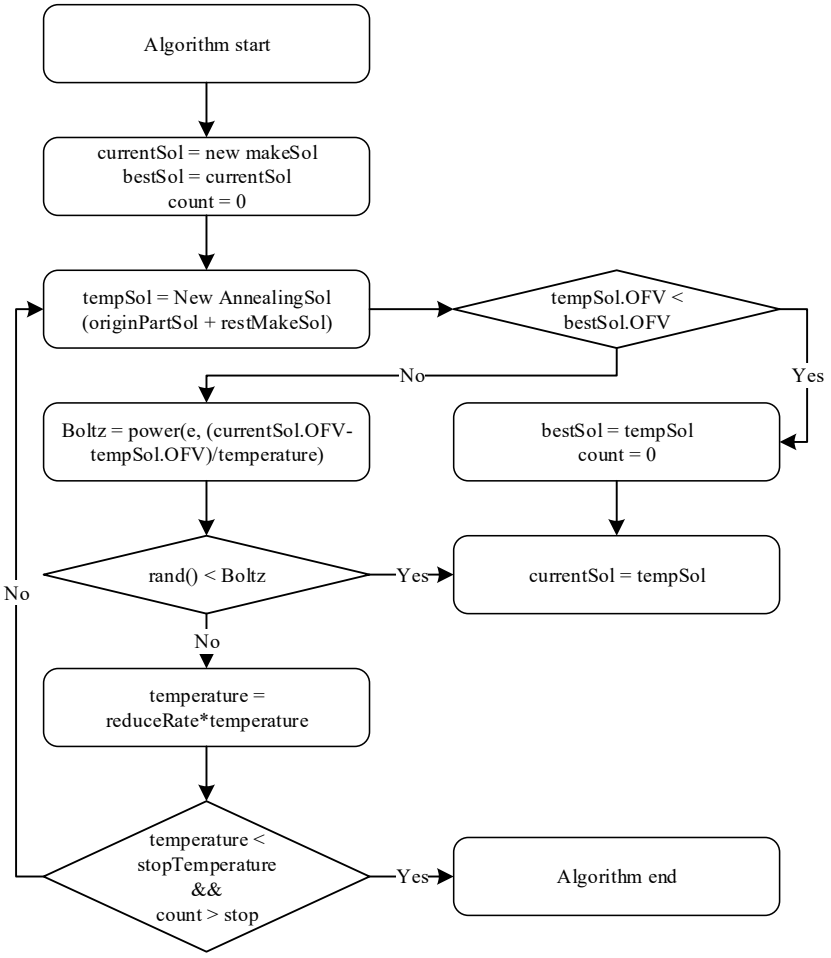


Figure 5. The flowchart of SA.

The flow of SA is shown in Figure 5. The algorithm has two different stop criteria; temperature and loop count. When a new solution in next loop fails the Boltzmann factor check, a pre-determined reduction rate reduces the temperature. When a new solution in next loop passes the Boltzmann factor check, the loop number is increased when the best solution is not updated. Again, as in Figure 4, the number is reset to zero when updating the optimal solution.

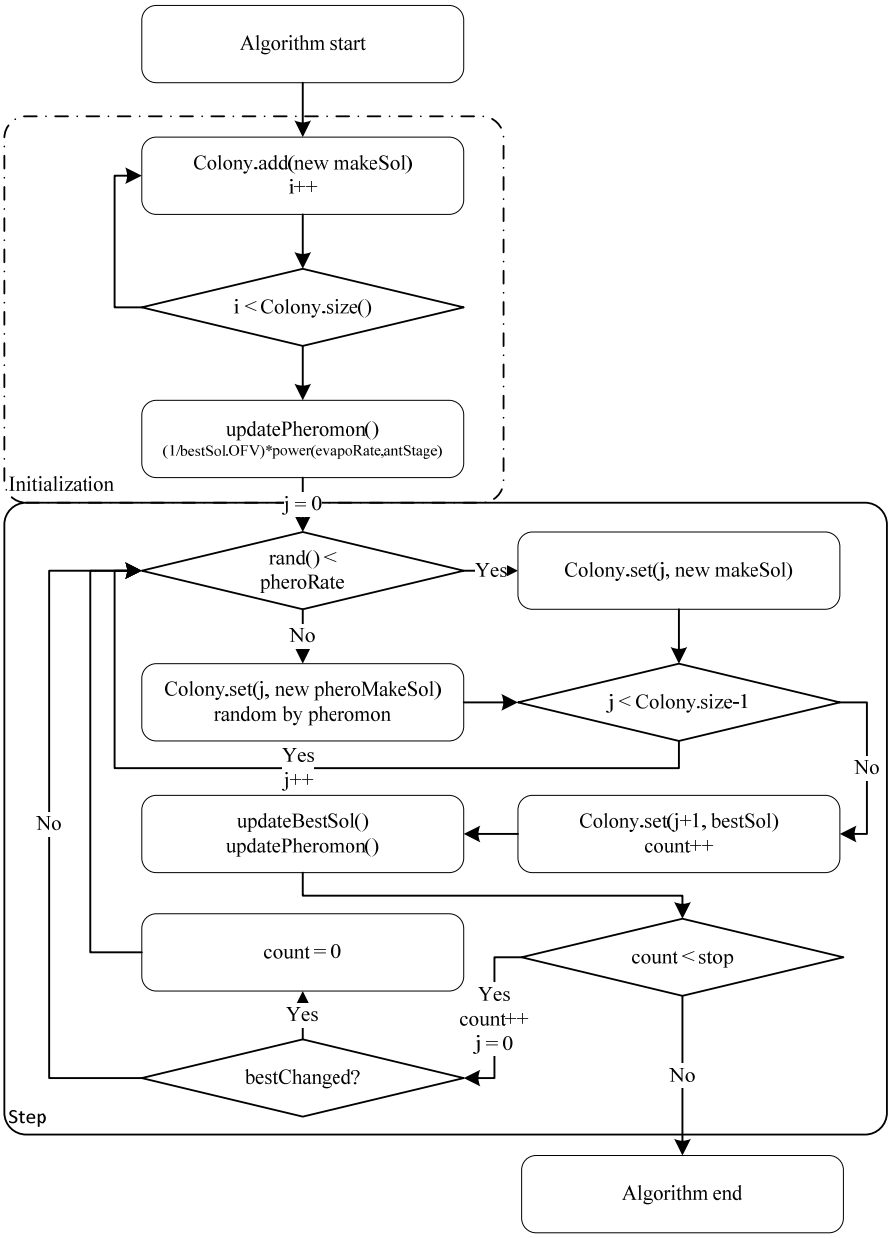


Figure 6. The flowchart of ACO.

ACO flow has two steps, initialization and improvement step, as shown in Figure 6. Ant colony is initialized in the initialization using the same method used with other algorithms. In the improvement step, a solution will be generated in one of two ways. The first way is a pheromone concept and the other is a reusing method in Initialization step. Each link has its own probability and ants find a path following the probability in order to make pheromone concept. The link probability is added by the following equation.

$$linkProbability(count+1) = linkProbability(count) + \frac{1}{bestOFV} evapoRate^{linkStep}, \quad (2)$$

the improvement step has a step count trigger for escaping the loop. The number is increased by one, when the flow pass a loop and it is set as zero, when the best solution is updated.

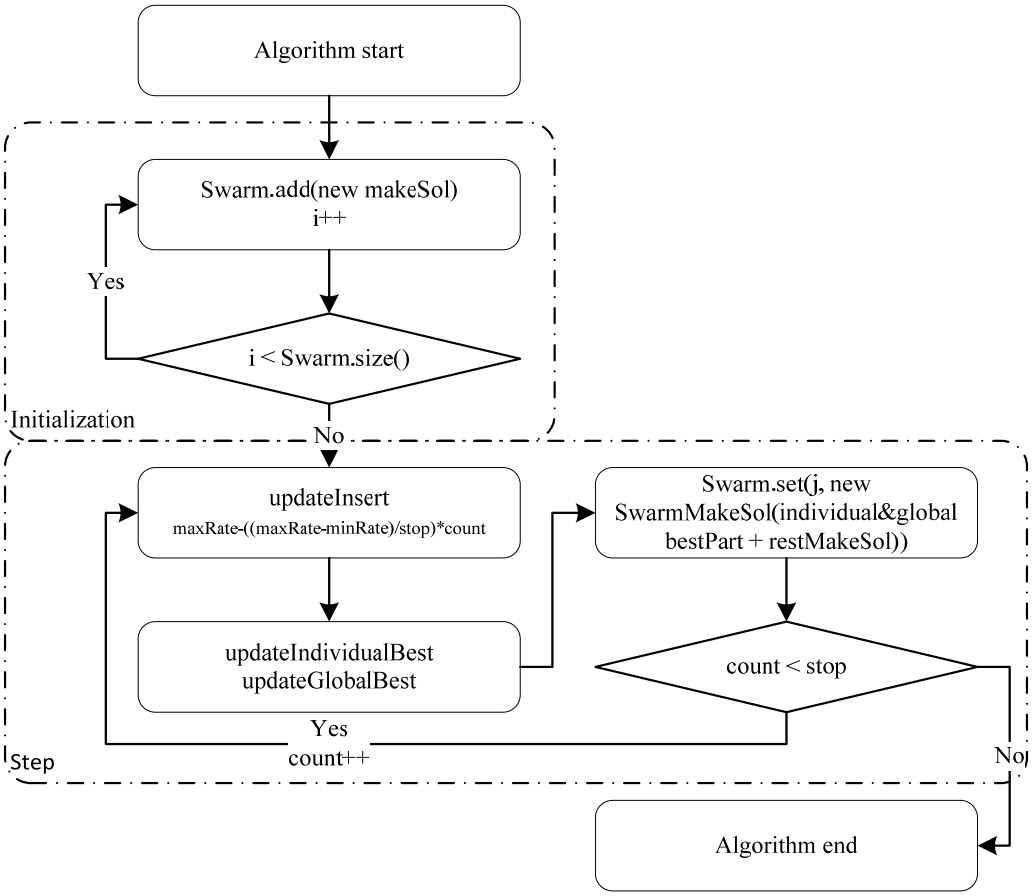


Figure 7. The flowchart of PSO.

In Figure 7, flow of PSO is introduced as two parts: initialization and improvement step. The algorithm has a similar structure to that of ACO. However, instead of concept of pheromone, PSO uses a concept of inertia rate for mimicking the movement of swarm. Previous solutions are destroyed by the inertia rates and new solutions are rebuilt using the best solution and an overlapped method for making a new solution. A number for stopping criteria is counted by passing the loop and the number is set as zero, when the best global solution is updated.

A method for making a new initial solution influences the performance of the algorithms. Thus, in Figure 4–7, the algorithms use the same method, called 'makeSol' in order to limit its effects on the algorithms. The flow of the method which includes a concept of potential field attempting to connect to the goal point is shown in Figure 8.

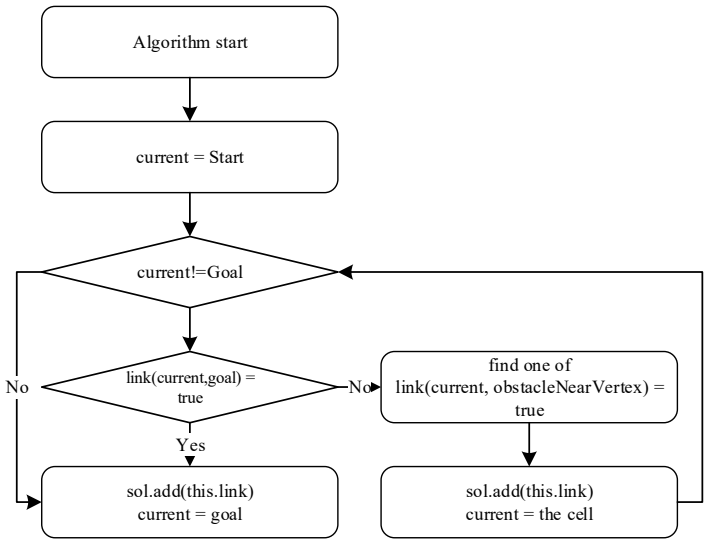


Figure 8. The flowchart of 'makeSol' logic.

3.2. Enviornments

In order to test the performance under several different environments, this research uses typical environments. In this paper, three different environment sizes are used for the experiments. **Figure 9** shows six environments which have 10 × 10 cells. These environments are formed archetypal patterns, such as 'n' shape, rectangles and slots. Environments, which have 'n' shape obstacle, are presented in Figure 9a, b, d. Figure 9c is a simple environment with squared obstacle and Figure 9e has slot obstacles which are known as difficult problem. The obstacle in Figure 9f is a part of environment in the paper by Karaman [44].

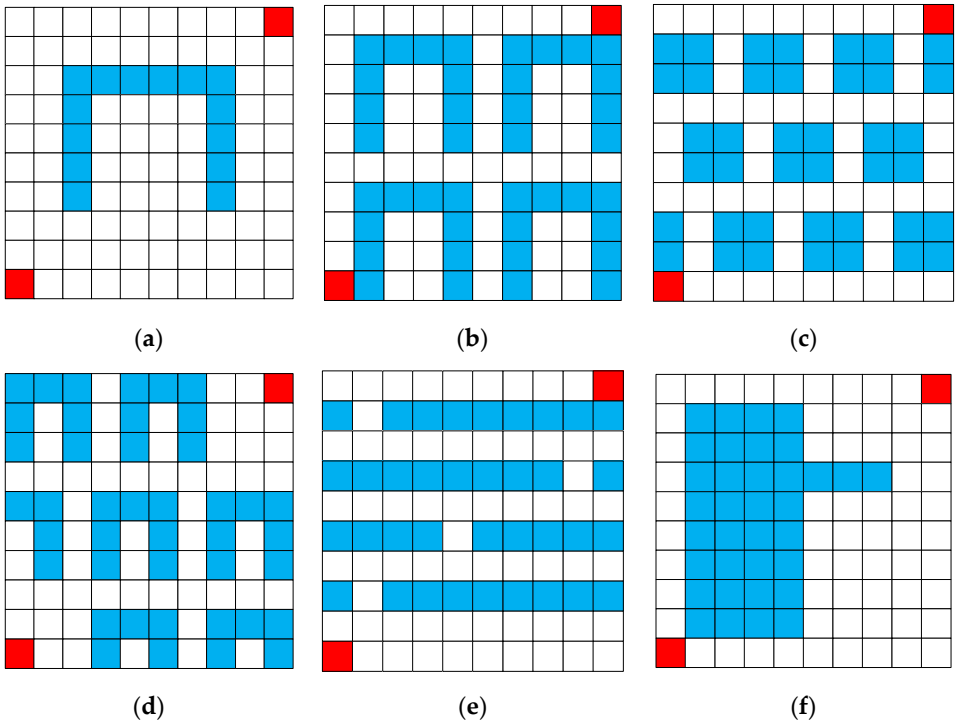
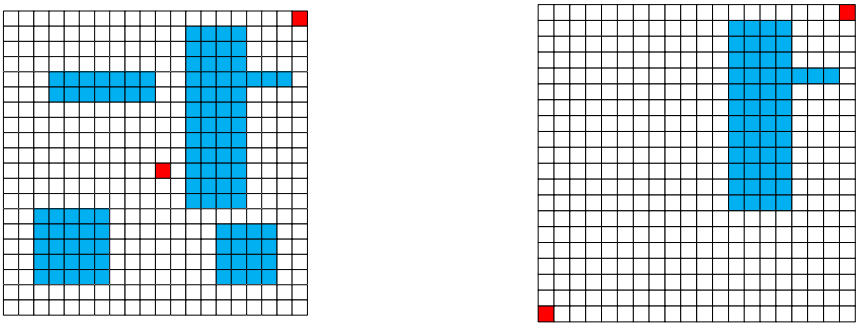


Figure 9. The experimental configuration of the 10 × 10 cells: (**a**) 10_1n; (b) 10_4n; (**c**) 10_11s; (**d**) 10_7n; (**e**) 10_4ch; and, (**f**) 10_1o.

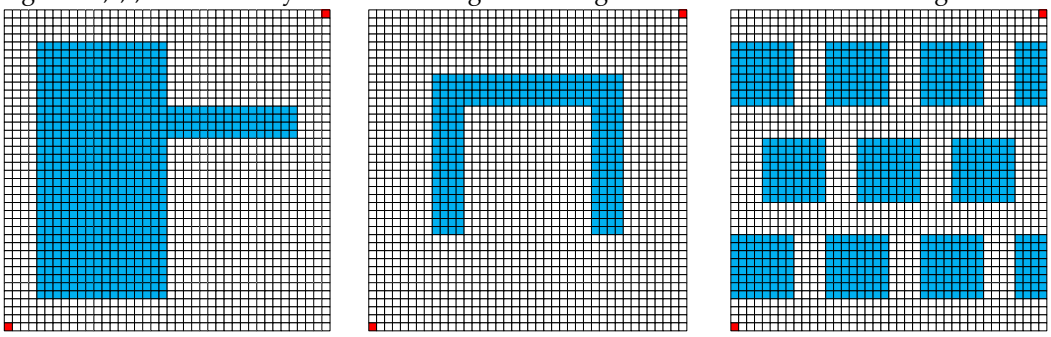
Two environments with 20 × 20 cells are presented on Figure 10. These environments also are reconfigured from the environment which is introduced in the paper by Karaman. An environment shown in Figure 10a is reproduced from the origin figure and Figure 10b is reconstituted based on the environment.



(**a**) (**b**)

Figure 10. The experimental configuration of the 20×20 cells: (**a**) 20_4o; (**b**) 20_1o.

Three environments with 40 × 40 girds are presented in in Figure 11. in order to measure the different performance by network size. The environments have same shape as the environments in Figure 9a,c,f, with the only difference being that the figures are divided into smaller grids.



(a) (b) (c)

Figure 11. The experimental configuration of the 40 × 40 cells: (**a**) 40_1o; (**b**) 40_1n; and, (**c**) 40_11s.

3.3. Metaheuristic Parameter

Metaheuristic algorithms are affected by their parameters. This is a reason why parameters must be carefully defined. The parameters used in this paper are determined by each searching algorithm and they are shown in Table 8. This research controlled two styles of parameters: population (ant and particle) number and stop criteria. The population number is set as 10 and stop criteria is set as 100.

Table 8. Tuning range of the parameters.

Searching algorithm	Parameter	Characteristic
GA	Population Number	Integer
	Stop Criteria	Integer
	Mutate Rate	Double (0~1)
	Crossover Rate	Double (0~1)
	Random Rate	Double (0~1)
SA	Stop Criteria	Integer
	Temperature	Double
	Reduce Rate	Double (0~1)
	Stop Temperature	Double
ACO	Ant Number	Integer
	Stop Criteria	Integer
	Pheromone Rate	Double (0~1)

	Evaporate Rate	Double (0~1)
PSO	Particle Number	Integer
	Stop Criteria	Integer
	Inertia Max Rate	Double (0~1)
	Inertia Min Rate	Double (0~1)

Many experimental design techniques are used, including full factorial experimental design, in order to tune the algorithm parameters. In this paper, one of the packages in R, named 'irace', is used to set the algorithm parameters [122]. The software package makes automatic sampling configurations and measures the configurations while using Friedman's test or t-test. Additionally, it is designed to redefine the distribution of the sampling and the package iterates this flow. The tuning procedure is performed for environment sets with a budget of 2000 experiments and the initial parameter ranges are defined, as follows: $Mutation\ Rate \in [0.01, 0.99]$; $Crossover\ Rate \in [0.01, 0.99]$; $Random\ Rate \in [0.01, 0.20]$; $Temperature \in [1000, 10000]$; $Reduction\ Rate \in [0.50, 0.95]$; $Stopping\ Temperature \in [0.01, 1.00]$; $Pheromone\ Rate \in [0.01, 0.50]$; $Evaporate\ Rate \in [0.50, 0.99]$; $Inertia\ Max\ Rate \in [0.50, 0.99]$; $Inertia\ Min\ Rate \in [0.01, 0.50]$. Each configuration is measured total summation of all environments objective function value. Table 9 shows the result of the tuned parameter.

Table 9. Tuned value of the parameters.

Searching algorithm	Parameter	Value
GA	Mutate Rate	0.97
	Crossover Rate	0.59
	Random Rate	0.15
SA	Temperature	2625.31
	Reduction Rate	0.61
	Stopping Temperature	0.52
ACO	Pheromone Rate	0.44
	Evaporate Rate	0.96
PSO	Inertia Max Rate	0.94
	Inertia Min Rate	0.36

4. Computational Results

Using the tuned parameter by 'irace', this paper provides performance of the algorithms through repeatability. The experiment is run 1000 times under the same computational environment, with the following specifications: Intel Core i9-7900X central processing unit (CPU) (3.30GHz) processor; 32 GB of memory; and, Windows 10. The algorithm is developed using Java jdk11.0.1. The used environments for the experiment are three sizes are and they are shown in Figure 9–11. The results of the experiment are organized in following tables. In the table, 'processing time' means that the shortest consumed time of best OFV. The result is rounded off to three decimal places. Additionally, the notated number '0.0' on time means that measured time is less than 1 millisecond.

Table 10 and 11 show the results of experiment under 10×10 grid environments. The performances of metaheuristic algorithm are distinguished clearly, depending on obstacle type, the number of obstacles, and algorithms. SA takes a shorter time to solve the problem than population-based algorithms, which are GA, ACO, and PSO, but the solution quality of SA has larger fluctuations. In Table 10, some cases of reported time variances of GA and ACO are larger than others. This shows that the proposed GA and ACO have more unsettled performance than others in some cases. The algorithms have a weakness for a problem with a huge local optimum, termed the 10_4ch problem. Additionally, the larger number of obstacles, which are 10_7n and 10_11s problems, decreases the algorithm performance.

Table 10. Experimental results of the metaheuristic algorithms within 10×10 case.

Problem name	Searching algorithm	Best OFV	Mean OFV	Processing time (ms)	Average processing time (ms)	OFV variance	Time variance

10_1o	GA	14.817	14.820	0.0	14.661	0.001	28.056
	SA	14.817	15.745	0.0	0.420	1.726	6.370
	ACO	14.817	14.817	0.0	7.567	0	61.731
	PSO	14.817	14.817	0.0	10.043	0	60.053
10_4ch	GA	27.136	29.893	281	355.175	1.807	14129.98
	SA	27.648	29.889	0.0	8.135	1.819	62.096
	ACO	28.307	31.933	328	315.228	1.752	4752.647
	PSO	27.648	29.889	47	45.607	0.566	37.096
10_4n	GA	16.243	17.337	47	53.248	0.048	139.240
	SA	16.243	17.663	0.0	0.860	0.103	12.733
	ACO	16.243	17.355	31	17.06	0.034	25.762
	PSO	16.243	17.169	15	15.726	0.095	18.089
10_7n	GA	14.065	14.305	62	90.302	0.025	833.630
	SA	14.065	14.566	0.0	2.990	0.100	38.428
	ACO	14.055	14.351	47	41.365	0.040	202.931
	PSO	14.065	14.278	15	21.607	0.015	61.548
10_11s	GA	14.485	16.810	125	180.398	0.426	3084.414
	SA	14.485	17.652	15	7.544	0.871	62.567
	ACO	14.485	17.044	93	78.178	0.298	650.778
	PSO	14.485	17.119	47	52.074	0.296	63.746
10_1n	GA	14.777	14.844	0.0	17.058	0.020	79.302
	SA	14.777	15.505	0.0	0.265	0.176	4.069
	ACO	14.777	14.777	0.0	14.406	0.000	31.709
	PSO	14.777	14.777	0.0	7.556	0	62.998

The algorithms in Table 11 are heuristic algorithms, so each algorithm provides a unique solution. The algorithms are usually dominantly faster than metaheuristic algorithms when solving this size of problem. Wavefront algorithm is easily trapped in local optimum, such as 10_4ch, 10_4n, 10_7n, and 10_1n problem. However, the processing time of Wavefront algorithm is lower than other algorithms, except for the 10_4ch case. In this case, the algorithm cannot find a reasonable solution, because of local optimum. The algorithms find a unique and the best solution for each problem, as Dijkstra algorithm and A star algorithm function as exact algorithms with the condition of these experiments.

Table 11. Experimental results of the other algorithms within 10 × 10 case.

Problem name	Searching algorithm	Best OFV	Mean OFV	Processing time (ms)	Average processing time (ms)	OFV variance	Time variance
10_1o	Dijkstra	14.817	14.817	0.0	1.038	0	15.036
	A*	14.817	14.817	0.0	0.831	0	12.363
	Wavefront	17.414	17.414	0.0	0.189	0	2.946
10_4ch	Dijkstra	27.136	27.136	0.0	0.779	0	11.554
	A*	27.136	27.136	0.0	0.771	0	12.039
	Wavefront	219.782	219.782	15	15.932	0	4.964
10_4n	Dijkstra	16.243	16.243	0.0	0.815	0	12.133
	A*	16.243	16.243	0.0	0.621	0	9.275
	Wavefront	20.485	20.485	0.0	1.883	0	26.053
10_7n	Dijkstra	14.055	14.055	0.0	0.744	0	11.001
	A*	14.055	14.055	0.0	0.625	0	9.394
	Wavefront	17.314	17.314	0.0	0.031	0	0.481
10_11s	Dijkstra	14.485	14.485	0.0	1.123	0	16.288
	A*	14.485	14.485	0.0	0.763	0	11.322
	Wavefront	14.485	14.485	0.0	0.015	0	0.225
10_1n	Dijkstra	14.777	14.777	0.0	1.448	0	20.492
	A*	14.777	14.777	0.0	1.198	0	17.296
	Wavefront	72.083	72.083	0.0	0.300	0	4.655

Figure 12 and 13 show the best represented solutions while using the algorithm. Figure 12a shows the notation of the paths. Among the notations, 'others' means that all of the algorithms, except algorithms, which are shown individually in the figure. Some of them find the shortest path

and the weakness of wavefront algorithm is easily found. In case of Figure 12d, f, h and Figure 13d, the solutions of wavefront algorithm are trapped by obstacles, which has 'n' shapes or small slots.

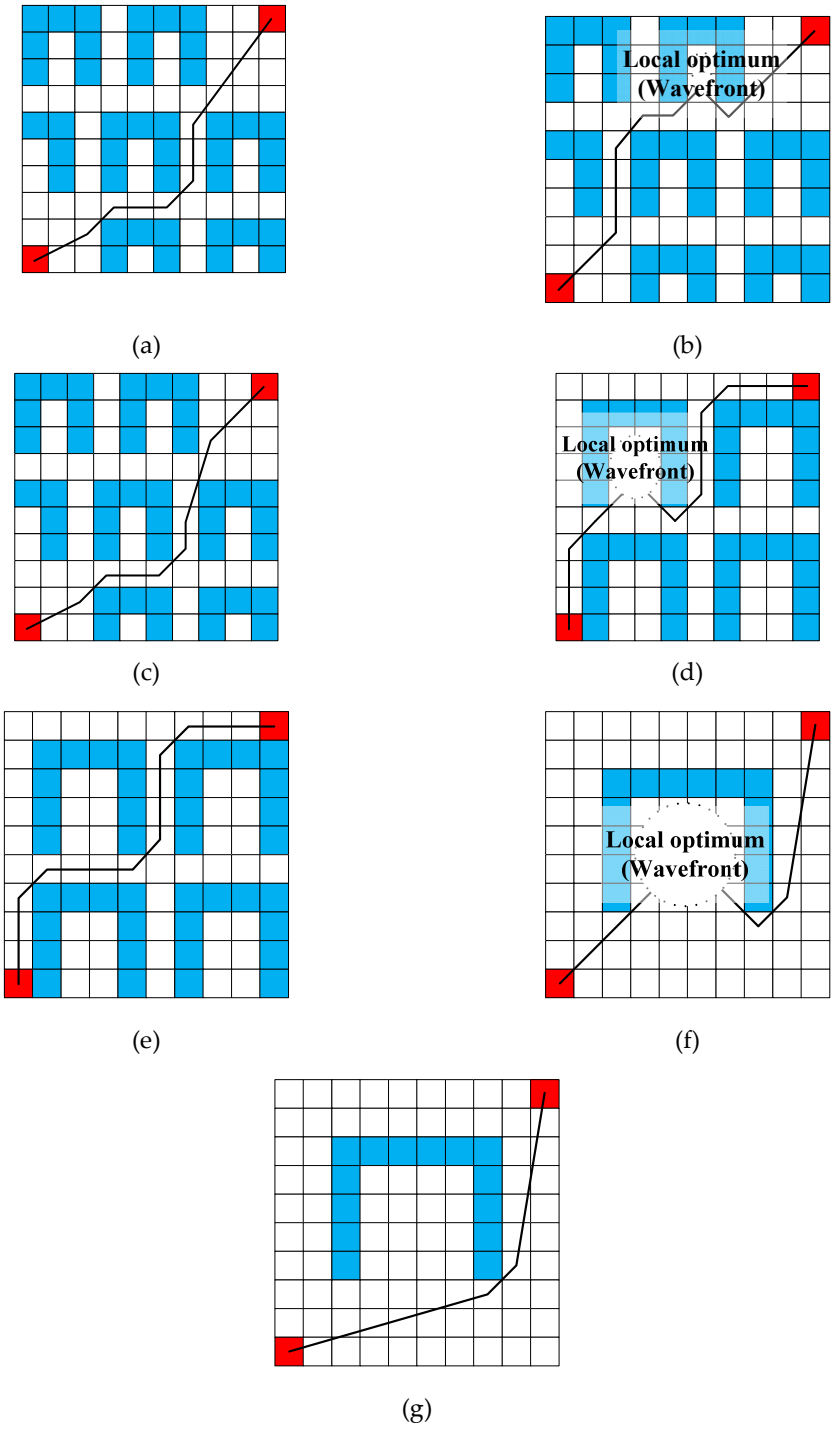


Figure 12. The best path of each algorithms with 'n' shape obstacles (**a**) 10_7n (GA, SA, PSO); (**b**) 10_7n (Wavefront); (**c**) 10_7n (Others); (**d**) 10_4n (Wavefront); (**e**) 10_4n (Others); (**f**) 10_1n (Wavefront); and, (**g**) 10_1n (Others).

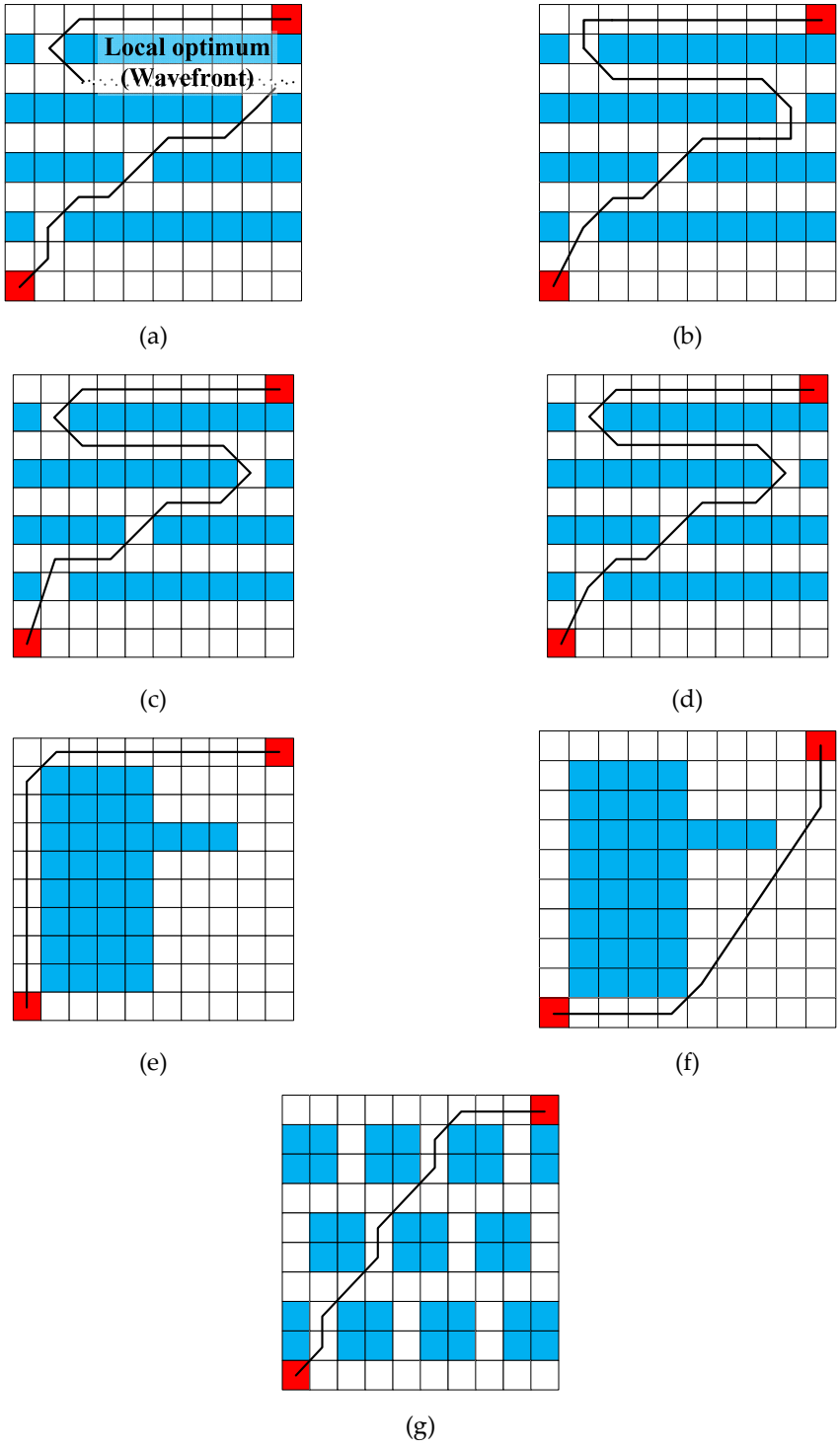


Figure 13. The best path with the rest environment (**a**) 10_4ch (Wavefront); (**b**) 10_4ch (ACO); (**c**) 10_4ch (SA, PSO); (**d**) 10_4ch (Others); (**e**) 10_1o (Wavefront); (**f**) 10_1o (Others); and, (**g**) 10_11s (all algorithms).

The results with 20 × 20 cell environments are organized in Table 12 and 13. Average processing time of Dijkstra algorithm and A star algorithm is recognizably increased in comparison with the results of 10 × 10 grid environments. The value, which is OFV variance of 20_4o with SA, is noticeably larger than other OFV variance. On the other hand, the other, the processing time, is only marginally different. For example, wavefront algorithm has a similar level with the outcome from

the smaller problem. Metaheuristic algorithms are still slower than other algorithms in this environment size and their average processing time have similar characteristics with the results of 10 × 10 grid environments.

Table 12. Experimental results of the metaheuristic algorithms within 20 × 20 case.

Problem name	Searching algorithm	Best OFV	Mean OFV	Processing time (ms)	Average processing time (ms)	OFV variance	Time variance
20_1o	GA	29.509	29.642	31	69.416	0.039	958.536
	SA	29.509	29.867	0.0	0.464	0.040	6.976
	ACO	29.509	29.509	140	186.123	0	238.633
	PSO	29.509	29.509	0.0	8.251	0	65.400
20_4o	GA	17.470	17.749	94	190.572	0.087	6469.725
	SA	17.470	18.497	0.0	2.137	3.040	28.829
	ACO	17.470	17.557	156	234.829	0.041	3034.65
	PSO	17.470	17.495	15	36.399	0.013	62.066

Table 13. Experimental results of the other algorithms within 20 × 20 case.

Problem name	Searching algorithm	Best OFV	Mean OFV	Processing time (ms)	Average processing time (ms)	OFV variance	Time variance
20_1o	Dijkstra	29.509	29.509	31	45.066	0	43.279
	A*	29.509	29.509	15	33.730	0	38.918
	Wavefront	30.971	30.971	0.0	0.328	0	5.025
20_4o	Dijkstra	17.470	17.470	15	23.931	0	61.980
	A*	17.470	17.470	0.0	8.360	0	62.405
	Wavefront	17.828	17.828	0.0	0.315	0	4.871

Table 14 and 15 show the experiment results under 40 × 40 cell. On this size of network, the searching speed of metaheuristic algorithms under some environments is faster than the speed of the Dijkstra algorithm and A star algorithm. For instance, PSO is a faster method for problems named 40_1o, 40_1n in comparison with the Dijkstra and A star algorithm. The best OFV of these experiments is the same as the outcome of the exact algorithm, which means that the OFV is the global optimum. The Wavefront algorithm is the fastest algorithm among these seven algorithms, but it is not able to escape from the local optimum, such as the 40_1n case. Meanwhile, in the case of 40_11s, the average processing time of GA and ACO significantly shows the weakness of the algorithms. The number of obstacles make their processing times increase when compared to 40_1o and 40_1n. Time variance is also unstable in the case with GA and ACO.

Table 14. Experimental results of the metaheuristic algorithms within 40 × 40 case.

Problem name	Searching algorithm	Best OFV	Mean OFV	Processing time (ms)	Average processing time (ms)	OFV variance	Time variance
40_1o	GA	62.512	62.622	125	156.54	0.037	2315.346
	SA	62.512	66.927	0.0	0.374	21.240	5.700
	ACO	62.512	62.512	515	546.614	0	108.868
	PSO	62.512	62.512	0.0	9.711	0	57.515
40_1n	GA	62.201	62.420	93	188.511	0.067	5195.051
	SA	62.201	63.481	0.0	0.327	1.619	4.995
	ACO	62.201	62.208	750	864.01	0.003	5275.087
	PSO	62.201	62.201	0.0	13.239	0.000	34.564
40_11s	GA	61.160	65.949	3359	2819.886	7.753	966961.7
	SA	60.677	68.213	63	23.476	15.223	116.924
	ACO	61.172	67.476	1250	1562.399	6.878	251384.9
	PSO	61.159	65.818	94	114.755	5.893	193.761

Table 15. Experimental results of the other algorithms within 40 × 40 case.

Problem name	Searching algorithm	Best OFV	Mean OFV	Processing time (ms)	Average processing time (ms)	OFV variance	Time variance
40_1o	Dijkstra	62.512	62.512	328	352.055	0	68.236
	A*	62.512	62.512	124	133.200	0	62.801
	Wavefront	73.899	73.899	0.0	1.486	0	21.079
40_1n	Dijkstra	62.201	62.201	609	625.265	0	47.522
	A*	62.201	62.201	390	407.943	0	39.447
	Wavefront	2727.895	2727.895	15	23.186	0	91.761
40_11s	Dijkstra	58.835	58.835	234	247.047	0	181.048
	A*	58.835	58.835	78	89.608	0	80.601
	Wavefront	64.527	64.527	0.0	1.276	0	18.266

Average processing time of these two algorithms is dramatically increased. In order to recognize the time changes clearly, Figure 14 is presented. The graph is the result of following equation.

$$f(x) = \frac{40\ probs\ Average\ processing\ time}{10\ probs\ Average\ processing\ time}, \quad (3)$$

As shown in the graph, Dijkstra and A star algorithms are remarkably more sensitive for network size than other algorithms, implying that network size is the weakness of the algorithms.

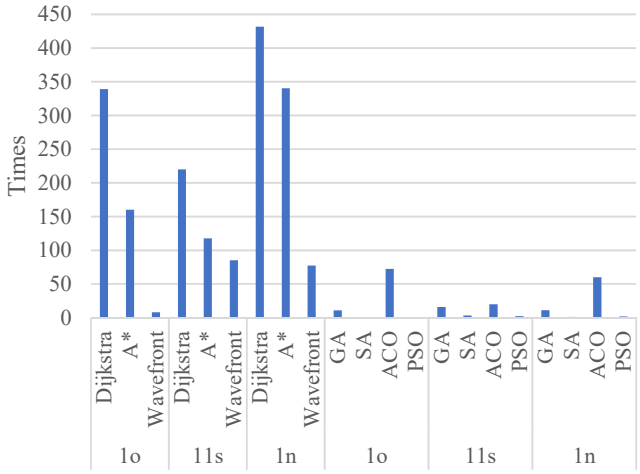


Figure 14. The process time increasing rate between 40 probs and 10 probs.

5. Conclusion

The problem of collision-free path planning has been studied in various areas resulting in many different criteria for studying the problem. These differences have given rise to several different terms with similar meanings, which, in turn, may actually hinder research for developing new methodologies for the problem of collision-free path planning. Meanwhile, many papers use varied searching algorithms, but their performances are not measured across the board under the exact same experimental conditions. The searching algorithm of research papers was selected from related works, because the algorithm is "typically used" and "good enough". For this reason, some papers provide performance reviews under the same categories, such as sampling-based algorithms. However, the papers do not show the differences between the exact algorithms and heuristic based algorithms at the same time. The algorithms each have their own advantages and disadvantages. Integrated research is required in order to measure the pros and cons. Thus, this paper has two main

contributions, which are categorization considering criteria and searching algorithms performance review under grid environments.

Firstly, this research has organized the problem while considering these criteria as four categories: environment type; environmental representation; searching algorithm; and experimental type. Each category has specific classes of characteristics and the difference in the characteristics make contribution of the other researches about CFPP problem. Through these categories, this research contributes to understand CFPP problem structure and recently studied topics.

Second, a performance review for seven different searching algorithms for a network environment are proposed in this study. Each algorithm has its own significant characteristics. The performance can be altered according to network size, feature of obstacles or number of obstacles. The experiments show that searching speed of Dijkstra and A star algorithm rapidly increases, depending on network size more so than metaheuristic algorithms. Performance of the metaheuristic algorithm fluctuates based on the number or shape of the obstacles. Therefore, this paper evaluated the performance of these searching algorithms.

Recently, CFPP problem is studied under dynamic, stochastic or multi tasks condition recently. However, the performance result of this paper focused on stationary obstacles, network model, and single task condition. Many other conditions, such as kinematic constraint, different environmental representation, or environment type, should be researched for measuring their effect. Also, metaheuristic algorithms are defined as simplified form, thus they may have a performance gap between the suggested algorithms and the studied algorithm recently.

We plan to redeem the limitation of this paper, as follows. Our further research will be studying performance of recent algorithms under various conditions using a car-like robot. Through this plan, various combined conditions of kinematic constraint, environmental representation, and environmental type can be expected to consider. Also, an actual robot has limited specifications, such as the limitation of sensing vague objects in real-time and computational limitations. With this further research, these limitations can be considered while using a car-like robot.

On the other hand, the extension of the algorithm can be adopted and applied to real-time environment specially in vision enabled mobile machines. The information captured by such vision would provide much more accurate information for a mobile agent to aid in avoiding collisions. Thus, future research could include the direction of path planning in a real time environment with a vision enabled system.

Author Contributions: Conceptualization, J.C, and H.S.; Data curation, H.S.; Formal analysis, H.S.; Funding acquisition, J.C.; Investigation, H.S.; Methodology, H.S.; Project administration, J.C.; Resources, H.S.; Software, H.S.; Supervision, J.C.; Validation, H.S.; Visualization, H.S.; Writing—original draft, H.S.; Writing—review & editing, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Korea Aerospace University faculty research grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Grazia Speranza, M. Trends in transportation and logistics. *Eur. J. Oper. Res.* **2018**, *264*, 830–836, doi:10.1016/j.ejor.2016.08.032.
2. Lozano-Pérez, T.; Wesley, M.A. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM* **1979**, *22*, 560–570, doi:10.1145/359156.359164.
3. Schwartz, J.T.; Sharir, M. On the "piano movers" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Commun. Pure Appl. Math.* **1983**, *36*, 345–398, doi:10.1002/cpa.3160360305.
4. Gasparetto, A.; Boscariol, P.; Lanzutti, A.; Vidoni, R. Trajectory Planning in Robotics. *Math. Comput. Sci.* **2012**, *6*, 269–279, doi:10.1007/s11786-012-0123-8.
5. Canny, J. *The Complexity of Robot Motion Planning*; The MIT Press: Cambridge, MA, USA, 1988; ISBN 9780262031363.
6. Sharir, M. Algorithmic motion planning in robotics. *Computer* **1989**, *22*, 9–19, doi:10.1109/2.16221.

7. Schwartz, J.T.; Sharir, M. A survey of motion planning and related geometric algorithms. *Artif. Intell.* . **1988**, *37*, 157–169, doi:10.1016/0004-3702(88)90053-7.
8. Hwang, Y.K.; Ahuja, N. Gross motion planning—A survey. *ACM Comput. Surv.* **1992**, *24*, 219–291, doi:10.1145/136035.136037.
9. Sariff, N.; Buniyamin, N. An overview of autonomous mobile robot path planning algorithms. In Proceedings of the 2006 4th Student Conference on Research and Development, Selangor, Malaysia, 27–28 June 2006; pp. 183–188.
10. Masehian, E.; Sedighizadeh, D. Classic and Heuristic Approaches in Robot Motion Planning–A Chronological Review. *World Acad. Sci. Eng. Technol.* **2007**, *29*, 101–106.
11. Goerzen, C.; Kong, Z.; Mettler, B. A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance. *J. Intell. Robot. Syst.* **2010**, *57*, doi:10.1007/s10846-009-9383-1.
12. Roberge, V.; Tarbouchi, M.; Labonte, G. Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning. *IEEE Trans. Ind. Inform.* **2012**, *9*, 132–141, doi:10.1109/TII.2012.2198665.
13. Mac, T.T.; Copot, C.; Tran, D.T.; De Keyser, R. Heuristic approaches in robot path planning: A survey. *Robot. Auton. Syst.* **2016**, *86*, 13–28, doi:10.1016/j.robot.2016.08.001.
14. Schwarting, W.; Alonso-Mora, J.; Rus, D. Planning and Decision-Making for Autonomous Vehicles. *Annu. Rev. Control Robot. Auton. Syst.* **2018**, *1*, 187–210, doi:10.1146/annurev-control-060117-105157.
15. Reif, J.H. Complexity of the mover's problem and generalizations. In Proceedings of the 20th Annual Symposium on Foundations of Computer Science (sfcs 1979), San Juan, Puerto Rico, USA, 29–31 October 1979; pp. 421–427, doi:10.1109/SFCS.1979.10.
16. Ó'Dúnlaing, C.; Yap, C.K. A "retraction" method for planning the motion of a disc. *J. Algorithms* **1985**, *6*, 104–111, doi:10.1016/0196-6774(85)90021-5.
17. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006; ISBN 9780511546877.
18. Mohanty, P.; Parhi, D. Controlling the Motion of an Autonomous Mobile Robot Using Various Techniques: A Review. *J. Adv. Mech. Eng.* **2013**, *1*, 24–39, doi:10.7726/jame.2013.1003.
19. Kavraki, L.E.; Kolountzakis, M.N.; Latombe, J.-C. Analysis of probabilistic roadmaps for path planning. *IEEE Trans. Robot. Autom.* **1998**, *14*, 166–171, doi:10.1109/70.660866.
20. Alexopoulos, C.; Griffin, P.M. Path planning for a mobile robot. *IEEE Trans. Syst. Man Cybern.* **1992**, *22*, 318–322, doi:10.1109/21.148404.
21. Latombe, J.-C. *Robot Motion Planning*; The Springer International Series in Engineering and Computer Science; Springer: New York, NY, USA, 2012; ISBN 9781461540229.
22. Barraquand, J.; Latombe, J.-C. A Monte-Carlo algorithm for path planning with many degrees of freedom. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; pp. 1712–1717.
23. Thrun, S.; Fox, D.; Burgard, W.; Dellaert, F. Robust Monte Carlo localization for mobile robots. *Artif. Intell.* **2001**, *128*, 99–141, doi:10.1016/S0004-3702(01)00069-8.
24. Janson, L.; Schmerling, E.; Pavone, M. Monte Carlo Motion Planning for Robot Trajectory Optimization Under Uncertainty. In *Robotics Research*; Springer: New York, NY, USA, 2018; pp. 343–361.
25. Lozano-Pérez, T. Spatial Planning: A Configuration Space Approach. *IEEE Trans. Comput.* **1983**, *32*, 108–120, doi:10.1109/TC.1983.1676196.
26. LaValle, S.M.; Branicky, M.S.; Lindemann, S.R. On the Relationship between Classical Grid Search and Probabilistic Roadmaps. *Int. J. Robot. Res.* **2004**, *23*, 673–692, doi:10.1177/0278364904045481.
27. Englot, B.; Hover, F. Multi-goal feasible path planning using ant colony optimization. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2255–2260.
28. Ichter, B.; Harrison, J.; Pavone, M. Learning Sampling Distributions for Robot Motion Planning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 7087–7094.
29. Kavraki, L.E.; Svestka, P.; Latombe, J.-C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580, doi:10.1109/70.508439.
30. Hsu, D.; Kindel, R.; Latombe, J.-C.; Rock, S. Randomized Kinodynamic Motion Planning with Moving Obstacles. *Int. J. Robot. Res.* **2002**, *21*, 233–255, doi:10.1177/027836402320556421.

31. Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA. Millennium Conference. In Proceedings of the IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; pp. 995–1001.
32. Kim, B.; Um, T.T.; Suh, C.; Park, F.C. Tangent bundle RRT: A randomized algorithm for constrained motion planning. *Robotica* **2016**, *34*, 202–225, doi:10.1017/S0263574714001234.
33. Malone, N.; Lesser, K.; Oishi, M.; Tapia, L. Stochastic reachability based motion planning for multiple moving obstacle avoidance. In Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control; Berlin, Germany, April 2014; pp. 51–60.
34. Kuwata, Y.; Karaman, S.; Teo, J.; Frazzoli, E.; How, J.P.; Fiore, G. Real-Time Motion Planning With Applications to Autonomous Urban Driving. *IEEE Trans. Control Syst. Technol.* **2009**, *17*, 1105–1118, doi:10.1109/TCST.2008.2012116.
35. Sánchez, G.; Latombe, J. On Delaying Collision Checking in PRM Planning: Application to Multi-Robot Coordination. *Int. J. Robot. Res.* **2002**, *21*, 5–26, doi:10.1177/027836402320556458.
36. Sanchez, G.; Latombe, J.-C. Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In Proceedings of the 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292), Washington DC, USA, 11–15 May 2002; pp. 2112–2119.
37. Sánchez, G.; Latombe, J.-C. A Single-Query Bi-Directional Probabilistic Roadmap Planner with Lazy Collision Checking. In Proceedings of the International Symposium on Robotics Research; Victoria, Australia, 9–12 November 2001; pp. 403–417.
38. Solovey, K.; Salzman, O.; Halperin, D. Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *Int. J. Robot. Res.* **2016**, *35*, 501–513, doi:10.1177/0278364915615688.
39. Wilmarth, S.A.; Amato, N.M.; Stiller, P.F. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In Proceedings of the 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C), Detroit, MI, USA, 10–15 May 1999; pp. 1024–1031.
40. Saha, M.; Latombe, J.-C.; Chang, Y.-C.; Prinz, F. Finding Narrow Passages with Probabilistic Roadmaps: The Small-Step Retraction Method. *Auton. Robot.* **2005**, *19*, 301–319, doi:10.1007/s10514-005-4748-1.
41. Saha, M.; Roughgarden, T.; Latombe, J.-C.; Sánchez-Ante, G. Planning Tours of Robotic Arms among Partitioned Goals. *Int. J. Robot. Res.* **2006**, *25*, 207–223, doi:10.1177/0278364906061705.
42. Dantam, N.T.; Kingston, Z.K.; Chaudhuri, S.; Kavraki, L.E. Incremental Task and Motion Planning: A Constraint-Based Approach. In Proceedings of the Robotics: Science and Systems; Ann Arbor, Michigan, USA, 18–22 June 2016.
43. Hsu, D.; Latombe, J.; Kurniawati, H. On the probabilistic foundations of probabilistic roadmap planning. *Int. J. Robot. Res.* **2006**, *25*, 627–643, doi:10.1177/0278364906067174.
44. Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. *Int. J. Robot. Res.* **2011**, *30*, 846–894, doi:10.1177/0278364911406761.
45. Marble, J.D.; Bekris, K.E. Asymptotically Near-Optimal Is Good Enough for Motion Planning. In *Robotics Research*; Springer: New York, NY, USA, 2017; pp. 419–436.
46. Tharwat, A.; Elhoseny, M.; Hassanien, A.E.; Gabel, T.; Kumar, A. Intelligent Bézier curve-based path planning model using Chaotic Particle Swarm Optimization algorithm. *Clust. Comput.* **2019**, *22*, 4745–4766, doi:10.1007/s10586-018-2360-3.
47. Bottasso, C.L.; Leonello, D.; Savini, B. Path Planning for Autonomous Vehicles by Trajectory Smoothing Using Motion Primitives. *IEEE Trans. Control. Syst. Technol.* **2008**, *16*, 1152–1168, doi:10.1109/TCST.2008.917870.
48. Yang, K.; Sukkarieh, S. An Analytical Continuous-Curvature Path-Smoothing Algorithm. *IEEE Trans. Robot.* **2010**, *26*, 561–568, doi:10.1109/TRO.2010.2042990.
49. Hsu, D.; Latombe, J.-C.; Motwani, R. Path planning in expansive configuration spaces. In Proceedings of the International Conference on Robotics and Automation, Albuquerque, NM, USA, April 1997; pp. 2719–2726.
50. Alterovitz, R.; Simeon, T.; Goldberg, K. The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty. In Proceedings of the Robotics: Science and Systems, Atlanta, GA, USA, 27–30 June 2007.

51. Contreras-Cruz, M.A.; Ayala-Ramirez, V.; Hernandez-Belmonte, U.H. Mobile robot path planning using artificial bee colony and evolutionary programming. *Appl. Soft Comput.* **2015**, *30*, 319–328, doi:10.1016/j.asoc.2015.01.067.
52. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numer. Math.* **1959**, *1*, 269–271, doi:10.1007/BF01386390.
53. Hart, P.E.; Nilsson, N.J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107.
54. Bhattacharya, P.; Gavrilova, M.L. Voronoi diagram in optimal path planning. In Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007), Glamorgan, UK, 9–11 July 2007; pp. 38–47.
55. Ho, Y.-J.; Liu, J.-S. Collision-free curvature-bounded smooth path planning using composite Bezier curve based on Voronoi diagram. In Proceedings of the 2009 IEEE International Symposium on Computational Intelligence in Robotics and Automation-(CIRA), Daejeon, South Korea, 15–18 Dec. 2009; pp. 463–468.
56. Janson, L.; Ichter, B.; Pavone, M. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *Int. J. Robot. Res.* **2018**, *37*, 46–61, doi:10.1177/0278364917714338.
57. Wang, H; Yu, Y; Yuan, Q. Application of Dijkstra algorithm in robot path-planning. In Proceedings of the 2011 Second International Conference on Mechanic Automation and Control Engineering, Hohhot, China, 15–17 July 2011; pp. 1067–1069.
58. Herman, M. Fast, Three-Dimensional, Collision-Free Motion Planning. In Proceedings of the 1986 IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 7–10 April 1986; pp. 1056–1063.
59. Bohlin, R.; Kavraki, L.E. Path planning using lazy PRM. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 24–28 April 2000; pp. 521–528.
60. van den Berg, J.; Ferguson, D.; Kuffner, J. Anytime path planning and replanning in dynamic environments. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, Orlando, FL, USA, 15–19 May 2006; pp. 2366–2371.
61. Noto, M.; Sato, H. A method for the shortest path search by extended Dijkstra algorithm. In Proceedings of the SMC 2000 Conference Proceedings, 2000 IEEE International Conference on Systems, Man and Cybernetics. "Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions" (Cat. No.00CH37166), Nashville, TN, USA, 8–11 Oct. 2002; pp. 2316–2320.
62. Deng, Y.; Chen, Y.; Zhang, Y.; Mahadevan, S. Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Appl. Soft Comput.* **2012**, *12*, 1231–1237, doi:10.1016/j.asoc.2011.11.011.
63. Duchoň, F.; Babinec, A.; Kajan, M.; Beňo, P.; Florek, M.; Fico, T.; Jurišica, L. Path Planning with Modified a Star Algorithm for a Mobile Robot. *Procedia Eng.* **2014**, *96*, 59–69, doi:10.1016/j.proeng.2014.12.098.
64. Fu, L.-C.; Liu, D.-Y. An efficient algorithm for finding a collision-free path among polyhedral obstacles. *J. Robot. Syst.* **1990**, *7*, 129–137, doi:10.1002/rob.4620070107.
65. Stoyanov, T.; Magnusson, M.; Andreasson, H.; Lilienthal, A.J. Path planning in 3D environments using the Normal Distributions Transform. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 Oct. 2010; pp. 3263–3268.
66. AL-Taharwa, I.; Sheta, A.; Al-Weshah, M. A Mobile Robot Path Planning Using Genetic Algorithm in Static Environment. *J. Comput. Sci.* **2008**, *4*, 341–344, doi:10.3844/jcssp.2008.341.344.
67. Cai, Z.; Peng, Z. Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. *J. Intell. Robot. Syst. Theory Appl.* **2002**, *33*, 61–71, doi:10.1023/A:1014463014150.
68. Yang, Q.; Yoo, S.-J. Optimal UAV Path Planning: Sensing Data Acquisition Over IoT Sensor Networks Using Multi-Objective Bio-Inspired Algorithms. *IEEE Access* **2018**, *6*, 13671–13684, doi:10.1109/ACCESS.2018.2812896.
69. Hu, Y; Yang, S.X. A knowledge based genetic algorithm for path planning of a mobile robot. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA '04,New Orleans, LA, USA, 26 April–1 May 2004; pp. 4350–4355.
70. Jiang, A.; Yao, X.; Zhou, J. Research on path planning of real-time obstacle avoidance of mechanical arm based on genetic algorithm. *J. Eng.* **2018**, 1579–1586, doi:10.1049/joe.2018.8266.

71. Elshamli, A.; Abdullah, H.A.; Areibi, S. Genetic algorithm for dynamic path planning. In Proceedings of the Canadian Conference on Electrical and Computer Engineering 2004 (IEEE Cat. No.04CH37513), Niagara Falls, Ontario, Canada, 2–5 May 2004; pp. 677–680.
72. Zhao, M.; Ansari, N.; Hou, E.S.H. Mobile manipulator path planning by a genetic algorithm. *J. Robot. Syst.* **1994**, *11*, 143–153, doi:10.1002/rob.4620110302.
73. MahmoudZadeh, S.; Yazdani, A.M.; Sammut, K.; Powers, D.M. . Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms. *Appl. Soft Comput.* **2018**, *70*, 929–945, doi:10.1016/j.asoc.2017.10.025.
74. Lee, H.-Y.; Shin, H.; Chae, J. Path Planning for Mobile Agents Using a Genetic Algorithm with a Direction Guided Factor. *Electronics* **2018**, *7*, 212, doi:10.3390/electronics7100212.
75. Lamini, C.; Benhlima, S.; Elbekri, A. Genetic Algorithm Based Approach for Autonomous Mobile Robot Path Planning. *Procedia Comput. Sci.* **2018**, *127*, 180–189, doi:10.1016/j.procs.2018.01.113.
76. Tuncer, A.; Yildirim, M. Dynamic path planning of mobile robots with improved genetic algorithm. *Comput. Electr. Eng.* **2012**, *38*, 1564–1572, doi:10.1016/j.compeleceng.2012.06.016.
77. Sedighi, K.H.; Ashenayi, K.; Manikas, T.W.; Wainwright, R.L.; Heng-Ming Tai Autonomous local path planning for a mobile robot using a genetic algorithm. In Proceedings of the Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753), Portand, OR, USA, 19–23 June 2004; pp. 1338–1345.
78. Tu, J; Yang, S.X. Genetic algorithm based path planning for a mobile robot. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Taipei, Taiwan, 14–19 Sept. 2003; pp. 1221–1226.
79. Nazarahari, M.; Khanmirza, E.; Doostie, S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Syst. Appl.* **2019**, *115*, 106–120, doi:10.1016/j.eswa.2018.08.008.
80. Janabi-Sharifi, F.; Vinke, D. Robot path planning by integrating the artificial potential field approach with simulated annealing. In Proceedings of the Proceedings of IEEE Systems Man and Cybernetics Conference-SMC, Le Touquet, France, 17–20 Oct. 1993; pp. 282–287.
81. Zhu, Q.; Yan, Y.; Xing, Z. Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing. In Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications, Jinan China, 16–18 Oct. 2006; pp. 622–627.
82. Park, M.G.; Jeon, J.H.; Lee, M.C. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In Proceedings of the 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No.01TH8570), Pusan, South Korea, 12–16 June 2001; pp. 1530–1535.
83. Janabi-Sharifi, F.; Vinke, D. Integration of the artificial potential field approach with simulated annealing for robot path planning. In Proceedings of the 8th IEEE International Symposium on Intelligent Control, Chicago, IL, USA, 25–27 Aug. 1993 pp. 536–541.
84. Park, M.G.; Lee, M.C. Experimental evaluation of robot path planning by artificial potential field approach with simulated annealing. In Proceedings of the Proceedings of the 41st SICE Annual Conference, Osaka, Japan, 5–7 Aug. 2002; pp. 2190–2195.
85. Miao, H.; Tian, Y.-C. Robot path planning in dynamic environments using a simulated annealing based approach. In Proceedings of the 2008 10th International Conference on Control, Automation, Robotics and Vision, Hanoi, Vietnam, 17–20 Dec. 2008; pp. 1253–1258.
86. Martínez-Alfaro, H.; Gómez-García, S. Mobile robot path planning and tracking using simulated annealing and fuzzy logic control. *Expert Syst. Appl.* **1998**, *15*, 421–429, doi:10.1016/S0957-4174(98)00055-4.
87. Tavares, R.S.; Martins, T.C.; Tsuzuki, M.S.G. Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning. *Expert Syst. Appl.* **2011**, *38*, 2951–2965, doi:10.1016/j.eswa.2010.08.084.
88. Miao, H.; Tian, Y.-C. Dynamic robot path planning using an enhanced simulated annealing approach. *Appl. Math. Comput.* **2013**, *222*, 420–437, doi:10.1016/j.amc.2013.07.022.
89. Kroumov, V.; Yu, J. 3D path planning for mobile robots using annealing neural network. In Proceedings of the 2009 International Conference on Networking, Sensing and Control, Okayama, Japan, 26–29 March 2009; pp. 130–135.

90. Carriker, W.F.; Khosla, P.K.; Krogh, B.H. The Use of Simulated Annealing to Solve the Mobile Manipulator Path Planning Problem. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; pp. 204–209.
91. Amer, H.M.; Al-Kashoash, H.; Hawes, M.; Chaqfeh, M.; Kemp, A.; Mihaylova, L. Centralized simulated annealing for alleviating vehicular congestion in smart cities. *Technol. Forecast. Soc. Chang.* **2019**, *142*, 235–248, doi:10.1016/j.techfore.2018.09.013.
92. Song, B.; Wang, Z.; Zou, L.; Xu, L.; Alsaadi, F.E. A new approach to smooth global path planning of mobile robots with kinematic constraints. *Int. J. Mach. Learn. Cybern.* **2019**, *10*, 107–119, doi:10.1007/s13042-017-0703-7.
93. Foo, J.L.; Knutzon, J.; Oliver, J.; Winer, E. Three-Dimensional Multi-Objective Path Planner for Unmanned Aerial Vehicles Using Particle Swarm Optimization. In Proceedings of the 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference; Honolulu, Hawaii, 23–26 April 2007; pp.1881
94. Zhang, Y.; Gong, D.; Zhang, J. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing* **2013**, *103*, 172–185, doi:10.1016/j.neucom.2012.09.019.
95. Fu, Y.; Ding, M.; Zhou, C. Phase Angle-Encoded and Quantum-Behaved Particle Swarm Optimization Applied to Three-Dimensional Route Planning for UAV. *IEEE Trans. Syst. Man Cybern. Part. A Syst. Hum.* **2011**, *42*, 511–526, doi:10.1109/TSMCA.2011.2159586.
96. Gong, D.; Zhang, J.; Zhang, Y. Multi-objective Particle Swarm Optimization for Robot Path Planning in Environment with Danger Sources. *J. Comput.* **2011**, *6*, 1554–1561, doi:10.4304/jcp.6.8.1554-1561.
97. Saska, M.; Macaš, M.; Přeučil, L.; Lhotská, L. Robot path planning using particle swarm optimization of ferguson splines. *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA* **2006**, *19*, 833–839, doi:10.1109/ETFA.2006.355416.
98. Zhang, Y.; Wu, L.; Wang, S. UCAV Path Planning by Fitness-Scaling Adaptive Chaotic Particle Swarm Optimization. *Math. Probl. Eng.* **2013**, doi:10.1155/2013/705238.
99. Chen, X.; Li, Y. Smooth Path Planning of a Mobile Robot Using Stochastic Particle Swarm Optimization. In Proceedings of the 2006 International Conference on Mechatronics and Automation, Luoyang, Henan, China, 25–28 June 2006; pp. 1722–1727.
100. Masehian, E.; Sedighizadeh, D. A multi-objective PSO-based algorithm for robot path planning. In Proceedings of the 2010 IEEE International Conference on Industrial Technology, Vina del Mar, Chile, 14–17 March 2010; pp. 465–470.
101. Wang, X.; Zhang, G.; Zhao, J.; Rong, H.; Ipate, F.; Lefticaru, R. A Modified Membrane-Inspired Algorithm Based on Particle Swarm Optimization for Mobile Robot Path Planning. *Int. J. Comput. Commun. Control.* **2015**, *10*, 732–745, doi:10.15837/ijccc.2015.5.2030.
102. Phung, M.D.; Quach, C.H.; Dinh, T.H.; Ha, Q. Enhanced discrete particle swarm optimization path planning for UAV vision-based surface inspection. *Autom. Constr.* **2017**, *81*, 25–33, doi:10.1016/j.autcon.2017.04.013.
103. Shiltagh, N.A.; Jalal, L.D. Optimal Path Planning For Intelligent Mobile Robot Navigation Using Modified Particle Swarm Optimization. *Int. J. Eng. Adv. Technol.* **2013**, *2*, 260–267.
104. Kang, H. Il; Lee, B.; Kim, K. Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm.In Proceedings of the 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, China, 19–20 Dec. 2008; pp. 1002–1004.
105. Alejo, D.; Cobano, J.A.; Heredia, G.; Ollero, A. Particle Swarm Optimization for collision-free 4D trajectory planning in Unmanned Aerial Vehicles. In Proceedings of the 2013 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 28–31 May 2013; pp. 298–307.
106. Thabit, S.; Mohades, A. Multi-Robot Path Planning Based on Multi-Objective Particle Swarm Optimization. *IEEE Access* **2019**, *7*, 2138–2147, doi:10.1109/ACCESS.2018.2886245.
107. Xing, L.N.; Rohlfshagen, P.; Chen, Y.W.; Yao, X. A Hybrid Ant Colony Optimization Algorithm for the Extended Capacitated Arc Routing Problem. *IEEE Trans. Syst. Man Cybern. Part. B (Cybern.)* **2011**, *41*, 1110–1123, doi:10.1109/TSMCB.2011.2107899.
108. Jiao, Z.; Ma, K.; Rong, Y.; Wang, P.; Zhang, H.; Wang, S. A path planning method using adaptive polymorphic ant colony algorithm for smart wheelchairs. *J. Comput. Sci.* **2018**, *25*, 50–57, doi:10.1016/j.jocs.2018.02.004.

109. Garcia, M.A.P.; Montiel, O.; Castillo, O.; Sepúlveda, R.; Melin, P. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Appl. Soft Comput.* **2009**, *9*, 1102–1110, doi:10.1016/j.asoc.2009.02.014.
110. Yen, C.-T.; Cheng, M.-F. A study of fuzzy control with ant colony algorithm used in mobile robot for shortest path planning and obstacle avoidance. *Microsyst. Technol.* **2018**, *24*, 125–135, doi:10.1007/s00542-016-3192-9.
111. Hsiao, Y.T.; Chuang, C.L.; Chien, C.C. Ant colony optimization for best path planning. In Proceedings of the IEEE International Symposium on Communications and Information Technology, Sapporo, Japan, 26–29 Oct. 2004; pp. 109–113.
112. Akka, K.; Khaber, F. Mobile robot path planning using an improved ant colony optimization. *Int. J. Adv. Robot. Syst.* **2018**, *15*, doi:10.1177/1729881418774673.
113. Yu, X.; Chen, W.-N.; Gu, T.; Yuan, H.; Zhang, H.; Zhang, J. ACO-A*: Ant Colony Optimization Plus A* for 3-D Traveling in Environments With Dense Obstacles. *IEEE Trans. Evol. Comput.* **2019**, *23*, 617–631, doi:10.1109/TEVC.2018.2878221.
114. Chia, S.H Su, K.L.; Guo, J.H.; Chung, C.Y. Ant Colony System Based Mobile Robot Path Planning. In Proceedings of the 2010 Fourth International Conference on Genetic and Evolutionary Computing, Shenzhen, China, 13–15 Dec. 2010; pp. 210–213.
115. Brand, M.; Masuda, M.; Wehner, N.; Xiao-Hua Yu Ant Colony Optimization algorithm for robot path planning. In Proceedings of the 2010 International Conference On Computer Design and Applications, Qinhuangdao, Chian, 25–27 June 2010.
116. Chao Zhang; Zhen, Z.; Daobo Wang; Meng Li UAV path planning method based on ant colony optimization. In Proceedings of the 2010 Chinese Control and Decision Conference, Xuzhou, China, 26–28 May 2010; pp. 3790–3792.
117. Cong, Y.Z.; Ponnambalam, S.G. Mobile robot path planning using ant colony optimization. In Proceedings of the 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Singapore, 14–17 July 2009; pp. 851–856.
118. Xiong, C.; Chen, D.; Lu, D.; Zeng, Z.; Lian, L. Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization. *Robot. Auton. Syst.* **2019**, *115*, 90–103, doi:10.1016/j.robot.2019.02.002.
119. Wang, L.; Kan, J.; Guo, J.; Wang, C. 3D Path Planning for the Ground Robot with Improved Ant Colony Optimization. *Sensors* **2019**, *19*, 815, doi:10.3390/s19040815.
120. Fan, Y.P.; Luo, X; Yi, S; Yang, S.Y.; Zhang, H. Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. In Proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing, Changsha, Hunan, China, 8–13 Oct. 2003; pp. 131–136.
121. Ma, Y.-N.; Gong, Y.-J.; Xiao, C.-F.; Gao, Y.; Zhang, J. Path Planning for Autonomous Underwater Vehicles: An Ant Colony Algorithm Incorporating Alarm Pheromone. *IEEE Trans. Veh. Technol.* **2019**, *68*, 141–154, doi:10.1109/TVT.2018.2882130.
122. López-Ibáñez, M.; Dubois-Lacoste, J.; Pérez Cáceres, L.; Birattari, M.; Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **2016**, *3*, 43–58, doi:10.1016/j.orp.2016.09.002.