Application of Metaheuristic Search Algorithms for Path Planning of Smart Vehicles

Osinachi Mbah

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

Eastern Mediterranean University
June 2023
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Mechanical Engineering.

Assoc. Prof. Dr. Murat Özdenefe
Chair, Department of Mechanical Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Mechanical Engineering.

Prof. Dr. Qasim Zeeshan
Supervisor

Examining Committee

1. Prof. Dr. Qasim Zeeshan

2. Assoc. Prof. Dr. Shaban Ismael Albrka

3. Asst. Prof. Dr. Omid Shekoofa

ABSTRACT

Navigating a smart vehicle in an environment, determined or unknown, requires the localization of such vehicle in that environment using GPS, cameras, vision, laser or ultrasonic sensors, motion planning of the vehicle in free configuration space of the environment, and its ability to deviate from obstacles. In planning a path from a start to a goal configuration, the aim is to obtain the shortest path in less time while avoiding obstacles. Metaheuristic algorithms have been extensively applied to achieve this; while it exploits the environment from an initial solution, it also explores it to find a possible feasible path. Researchers in robotics and automation field have investigated and analysed the performance of population-based algorithms like Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and Cuckoo Search Algorithm (CSA) to obtain a feasible shortest path. This research work investigates the performance of metaheuristic search algorithms like GA, PSO, FA, and CSA for path planning on four different benchmark problem maps (40 x 40m large, 20 x 20m maze, 20 x 20m rockpile and 20 x 20m pothole) and makes a comparative analysis based on computational time and path length. Furthermore, three sampling methods i.e., Random, Latin hypercube and pseudo-uniform sampling were used. It is observed that all the algorithms were able to achieve the optimal, although CSA performed poorly on path distance using GA and PSO on the maze and pothole map respectively, its performance on other two maps was relatively better both on shortest distance and computational time, however, it is concluded that no single algorithm is universally the best-performing algorithm for all maps. All simulations were performed on MATLAB R2022b.

Keywords: Metaheuristic Algorithms, Path Planning, Navigation, Mobile Robot, Smart Vehicle

ÖZ

Akıllı bir aracın belirli veya bilinmeyen bir ortamda yönlendirilmesi, GPS, kameralar, görüntü, lazer veya ultrasonik sensörler kullanılarak aracın o ortamdaki yerinin belirlenmesini, aracın ortamın serbest konfigürasyon alanında hareket planlamasının yapılmasını ve engellerden sapmak. Başlangıçtan hedef konfigürasyonuna kadar bir yol planlarken amaç, engellerden kaçınarak en kısa yolu daha kısa sürede elde etmektir. Metasezgisel algoritmalar, bunu başarmak için kapsamlı bir şekilde uygulanmıştır; çevreyi bir başlangıç çözümünden yararlanırken, aynı zamanda olası bir uygulanabilir yol bulmak için de araştırır. Robotik ve otomasyon alanındaki araştırmacılar, Genetik Algoritma (GA), Karınca Kolonisi Optimizasyonu (ACO), Parçacık Sürü Optimizasyonu (PSO), Ateşböceği Algoritması (FA) ve Cuckoo Arama Algoritması (CSA) gibi popülasyon tabanlı algoritmaların performansını araştırdı ve analiz etti uygun bir en kısa yol elde etmek için. Bu araştırma çalışması, GA, PSO, FA ve CSA gibi metasezgisel arama algoritmalarının dört farklı kıyaslama problem haritası (40 x 40m büyük, 20 x 20m labirent, 20 x 20m kaya yığını ve 20 x 20m çukur) üzerinde yol planlaması için performansını araştırıyor ve hesaplama süresine ve yol uzunluğuna dayalı olarak karşılaştırmalı bir analiz yapar. Ayrıca, üç örnekleme yöntemi, yani Rastgele, Latin hiperküp ve sözde tekdüze örnekleme kullanılmıştır. Tüm algoritmaların optimumu yakalayabildiği, CSA'nın sırasıyla labirent ve çukur haritasında GA ve PSO kullanarak yol mesafesinde düşük performans göstermesine rağmen, diğer iki haritadaki performansının hem en kısa mesafe hem de hesaplama süresinde nispeten daha iyi olduğu görülmüştür. ancak, tek bir algoritmanın tüm haritalar için evrensel olarak en iyi performans gösteren algoritma olmadığı sonucuna varılmıştır. Tüm simülasyonlar MATLAB R2022b üzerinde gerçekleştirilmiştir.

Anahtar Kelimeler: Metasezgisel Algoritmalar, Yol Planlama, Navigasyon, Mobil Robot, Akıllı Araç

ACKNOWLEDGMENTS

I wish to register my profound gratitude to God for His mercy and grace bestowed on me throughout this master's program. I would love to really appreciate my parents and siblings for their profound support, prayers, and advice throughout the period of my study. My deepest gratitude goes to my supervisor, Prof. Dr. Qasim Zeeshan for his guidance and immense support right from the beginning of this thesis. His ideas and information has enriched my knowledge especially in my field.

I would love to acknowledge the efforts and support of Assoc. Prof. Dr. Babak Safaei, Assist. Prof. Dr Mohammed Asmael, Assoc. Prof. Dr. Murat Ozdenefe wih all the other instructors and the jury members (Assoc. Prof. Dr. Shaban Ismael Albrka, Assist. Prof. Dr Omid Shekoofa). I am very lucky for having very supportive colleagues and friends (Zommorod, Emmanuel, Gizem, Alibar, Yasser, Rana, Zahra, Erfan, Aysegul, Mert, Hussain). I am very grateful for the advice, guidance, and support they have given me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
BA	Bat Algorithm
CSA	Cuckoo Search Algorithm
FA	Firefly Algorithm
GA	Genetic Algorithm
GWO	Grey Wolf Optimization
MVO	Multi-Verse Optimizer
PSO	Particle Swarm Optimization
TS	Tabu Search
WOA	Whale Optimization Algorithm
NFL	No-Free-Lunch

.

Chapter 1

INTRODUCTION

Due to emerging technologies like big data, cloud computing, 5G construction [1], IoT [2], and artificial intelligence [3,4], smart vehicles have evolved to use these technologies to reduce human error while in motion and also to avoid traffic (self-driving vehicles), assist in logistics and various manufacturing processes in industries (automated guided vehicle), operate in rough and dangerous terrains (unmanned ground vehicles), and has applications in healthcare, homes, e-commerce (autonomous mobile robots).

Self-driving vehicles are incorporated with Light Detection and Ranging (LIDAR) sensors [5], cameras infused with deep learning algorithms [6], ultrasonic sensor [7] for vehicle and object detection, traffic alerts, detect zebra crossings, and avoid collision with humans. Automated guided vehicles (AGV) are used in warehouses for logistics and material handling processes. They use lasers, magnets, vision cameras or follow marked lines or wires for navigation. Recently, AI, specifically reinforcement learning has been an important tool in route planning of automated guided vehicles [8]. Autonomous mobile robots (AMR) have many use cases in our world today. They can be used for inspection [9], Surveillance [10], monitoring [11], logistics and service purposes [12–15]. Autonomous mobile robots can be classified as holonomic and non-holonomic [16,17]. Controllable degrees of freedom are equal to total degrees of freedom in holonomic mobile robots, i.e. they can move in all directions possible

within the configuration space. In the realm of mobile robotics, the abstraction known as the robot, or base, is referred to as a holonomic mobile robot without consideration for the rigid bodies that actually make up the mechanism. As a result, holonomic mobile robots are smart vehicles whose motion in a plane is in three degrees of freedom [18]. Non-holonomic mobile robots have their velocities and other derivatives of their position constrained [19].

Smart vehicles need to understand the nature of its environment so it can update its actions to increase the chance of reaching the goal position during navigation in the environment. The smart vehicle must provide the following information: Where am I? Where am I headed? How can I travel there? What obstacles must I overcome? Three fundamental elements - mapping, localization, path planning - answer these questions [20].

- Mapping: In mapping, the smart vehicles require a map of its environment. The map gives the locations and directions of such vehicle. This map can be built while the smart vehicle discovers the environment or can be stored its memory.
- Localization: It's important that smart vehicles know their location on the map. Cameras, GPS, sensors like laser, vision and ultrasonic sensors and the information obtained from them can be combined to localize such smart vehicle. The location can be expressed in absolute coordinates (longitude, latitude, altitude), denoted as a reference relative to the environment or indicated as topographical coordinate (e.g. in the sitting room).
- Path planning: Path planning involves determining an obstacle-free path through a given environment, which in real life is typically congested [21].

1.1 Problem Statement

With the existence of obstacles in an environment, finding the quickest direction of motion for smart vehicles becomes a problem. The task of determining an approach which will yield the shortest path comes into play. Due to the competitiveness of metaheuristic algorithms against other path planning approaches, it offers a greater chance of finding the global feasible shortest path [22]. Performing a comparative analysis on various metaheuristics is important to finding one that achieves the determined objective. Although, sometimes a path is planned around present obstacles [23], an acceptable path, to some degree shouldn't lean much on the edges of obstacles. Having sampled points on free configurations of the map, it is required to have a connectivity of points which define paths away from obstacles.

1.2 Objective and Novelty of this Study

The aim of the study is to perform a comparative analysis of various metaheuristic algorithms based on path distance and planning time. To highlight the novelty of this work, some sampling approaches never used in this field of study are incorporated. The sampling methods are random, latin hypercube and pseudo-uniform sampling, although random sampling has been considered in some literatures highlighted in chapter 2. Also, newly modified versions of some metaheuristic algorithms (PSO, CSO, FA) considered in this study are implemented. These modified versions arise from the need to have discrete metaheuristics which fits into this study compared to the basic versions that work with continuous values.

Chapter 2

LITERATURE REVIEW

2.1 Path Planning Methods and Algorithms for Smart Vehicles

Path planning can be considered as local or global. Global planning aims to search the optimum path given a huge amount of environmental data, and it works best when the environment is static and well-known to the smart vehicle. A complete pathway is generated by path planning algorithm from a starting to a destination point to produce the optimum trajectory for the smart vehicle. Local path planning is mostly applied in unfamiliar or changeable environments. While the smart vehicle is in motion, local path planning is performed using data from local sensors. The smart vehicle can generate a new route in response to dynamic changes in the environment. Various path planning methods and algorithms have been investigated by researchers in the robotic field. The kinematics, the dynamics of the environment, computation capabilities of smart vehicles, sensors used and other sourced information availability all influence which algorithm to choose. The trade-offs between performance and complexity of algorithm are also dependent on the use case [24]. Path planning methods and algorithms as shown in Figure 1, can be classified into classical methods like cell decomposition, metaheuristic algorithms like genetic algorithm, machine learning like reinforcement learning, sampling methods like probabilistic roadmap [22,24–26].

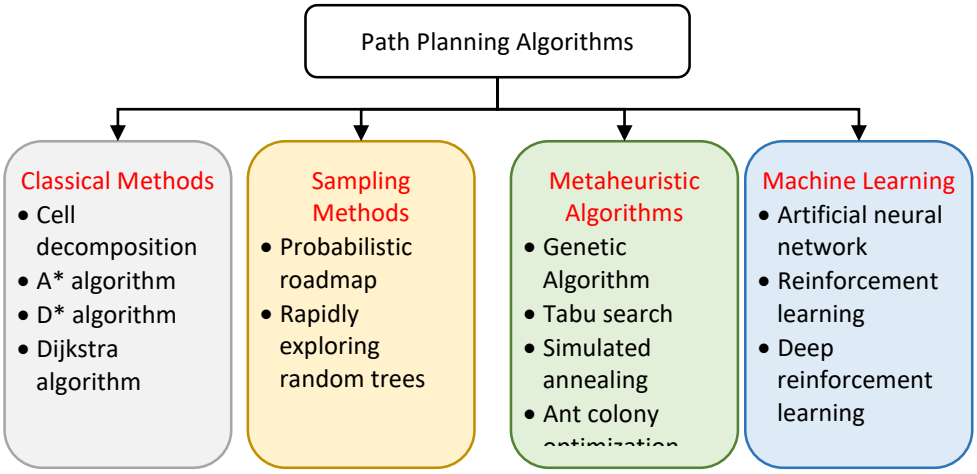


Figure 1: Classification of path planning algorithms

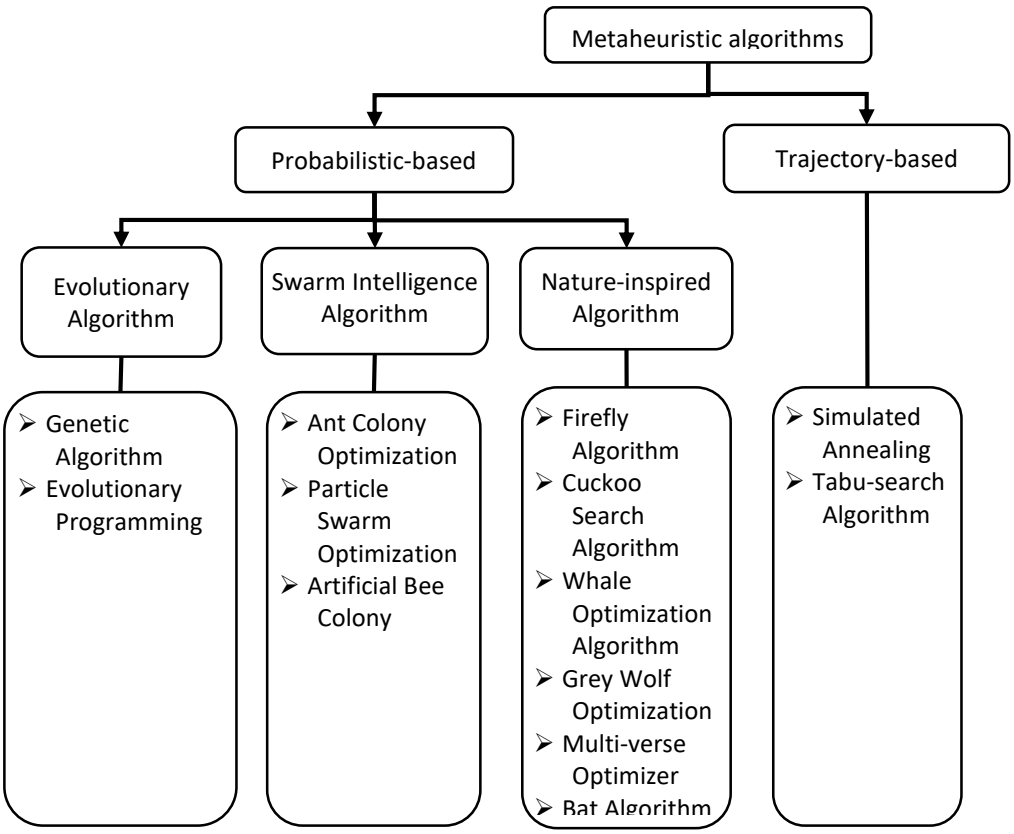


Figure 2: Classification of metaheuristic algorithms

2.2 Metaheuristic Algorithms

Metaheuristic algorithms are high-level heuristics that provide suitably solution to optimization problems, most especially problems with incomplete information or limited computational capacity. When implemented for path planning problems, they

are proficient in handling partially-known environments and environment with moving obstacles as compared to classical algorithms which require prior knowledge of the environment [22]. Metaheuristic algorithms are grouped into population-based like Particle Swarm Optimization and trajectory-based like Simulated Annealing as illustrated in Figure 2 [27]. While population-based metaheuristics generates sets of points in a search space, trajectory-based methods describe trajectory through a single point at each time-step in the search space during the search.

2.2.1 Genetic Algorithm (GA)

This optimization method applies genetics and natural selection as discovered first by Bremermann [28]. Holland first applied genetic algorithm to the field of computer science [29]. Its application can now be seen in robot navigation and all areas of science and technology. It is concerned with the optimization of complex problems in which the objective function value must be maximized or minimized under given boundaries. In this approach, the population size (chromosomes defined as sets of genes) is defined the given problem and a fitness value is given to every chromosome in the population depending upon the objective function. These chromosomes are selected for their fitness value and are allowed to pass on their genes to future generations via crossbreeding. Mutation maintains population diversity while preventing early convergence. Table A.1 describes the pseudocode of genetic algorithm. Finally, if the population has converged, the algorithm is terminated [30].

Much emphasis has been placed on GA-based methods in recent years due to their success in solving optimization problems, one of which is path planning; therefore, we believe GA can reasonably overcome it [6]. Table 1 outlines various studies on GA for path planning problem, lists the variables considered and remarks on some insights in

the study. The hybridization of GA with other intelligent algorithms such as GA-Fuzzy Logic [31], GA-Intelligent Water Drop [32] and GA-Neural Network [33], have been investigated by researchers to obtain better results. In using GA-based methods, distance is a common parameter [34–38] together with path smoothness and clearance [35,37,38], energy evaluation [39] and robot speed issues. [40] proposed improved Genetic algorithm to solve appointment order allocation and route planning problem of Cainiao unmanned vehicles. An optimized approach using genetic algorithm was proposed by [41] to implement the multi-objective evolutionary algorithm (MOEA) for planning a mobile robot's trajectory in a known environment. An experiment was performed with a two wheeled mobile robot localized ArUco system in robotic operating system (ROS). The proposed method is not suitable for rough terrains because the dynamics on the mobile robot is not considered. Because of the embedded system's low-end hardware, the proposed algorithm runs on a console computer.

Table 1: Genetic algorithm for path planning of smart vehicles

Types	Initial Population Generation Method	Population size	Reproduction Operators	Fitness Function	Sorting and Selection Technique	Number of generations	Type of vehicle	Type of Obstacle	Type of map	Software	Remarks	Ref
Improved GA	Random	100		$F = \frac{1}{C + MP}$	Optimization guidance factor and Roulette selection	500	Multiple	Static	Topological	MATLAB R2018a	• Order allocation and route planning problem is modelled to obtain efficient picking of orders. • Provides the optimal solutions to unmanned vehicle inputs and their path planning.	[40]

Types	Initial Population Generation Method	Population size	Reproduction Operators	Fitness Function	Sorting and Selection Technique	Number of generations	Type of vehicle	Type of Obstacle	Type of map	Software	Remarks	Ref
Novel GA	CBPRM	20, 25, 50	Crossover: Ordinary Mutation: Change, smooth and shortcut operators	C(k) = L(k) x S(k)		50	Single	Static	Geometrical	CGAL 3.3.1 [42]	• Minwoski sum is used as a common computational geometry based approach instead of cell based methods. • CBPRM speeds up the evolutionary process.	[43]
hTetro-GA	Random	25, 50, 100	Crossover: Single-point crossover operator	$F = \frac{1}{1+W_{A^*}(POS(p_{l_p}$-	NSGA-II technique.	50	Single (Reconfigurable tilted robot)	Static (H-Shaped, Spiral and 3-Slit)	24 x 24 Grid	Robotic Operating System (ROS)	• NSGA-II technique is implemented to	[44]

Types	Initial Population Generation Method	Population size	Reproduction Operators	Fitness Function	Sorting and Selection Technique	Number of generations	Type of vehicle	Type of Obstacle	Type of map	Software	Remarks	Ref
			Mutation: Classic GA mutation operator Removal Operator: Translational motion command, Non-translational motion command Rearrangement Operator					Dynamic (Perpendicular and Parallel)			• determine best motion command sequence. • hTetro-GA algorithm can be implemented for reconfigurable robots during a rescue task.	
Novel GA	Random	20	Crossover: Single-point with		Parent Selection: Rank-	2000	Single (Two wheeled mobile	Static	10 x 10 Grid	Robotic Operating	• Experimented mobile robot is	[41]

Types	Initial Population Generation Method	Population size	Reproduction Operators	Fitness Function	Sorting and Selection Technique	Number of generations	Type of vehicle	Type of Obstacle	Type of map	Software	Remarks	Ref
			crossover rate(pc) = 1.0 Mutation: Bitwise flipping with mutation rate(pm) = 0.1 (1/string length)		based roulette wheel Survivor selection: Elitism + crowding distance (NSGA-II)		robot running on STM32 microcontroller)			System (ROS) and Python 3	localized by the ArUco system through a bird's view camera. • Mobile robot can't operate on rough environment because its dynamic behaviour is not considered in the work.	

Types	Initial Population Generation Method	Population size	Reproduction Operators	Fitness Function	Sorting and Selection Technique	Number of generations	Type of vehicle	Type of Obstacle	Type of map	Software	Remarks	Ref
GA	Random		Crossover: single-point	$F = N - (\alpha_1 \sum_{i=1}^{n} d_i + \alpha_2 * m)$	Roulette	Between 100 and 500 generations	Single	Static (Special, Regular, Irregular multiple).	100 x 100 Geometrical		• Large turning angle problem in basic genetic algorithm is overcome. • Genetic algorithm implementation is separate from path smoothing process. • Experimental results are compared with	[45]

Types	Initial Population Generation Method	Population size	Reproduction Operators	Fitness Function	Sorting and Selection Technique	Number of generations	Type of vehicle	Type of Obstacle	Type of map	Software	Remarks	Ref
											Dijkstra algorithm.	

NSGA-II: Non-dominated sorting genetic algorithm-II; CBPRM: Clearance Based Probabilistic Roadmap Method; hTetro-Ga: hinged-Tetromino-Genetic Algorithm

2.2.2 Ant Colony Optimization (ACO)

M. Dorigo proposed the ant colony algorithm in his Ph. D. dissertation in 1992 [46]. Ants exhibit complex social behaviour when searching for food, owing to the pheromones they deposit. Pheromones draw in other ants and create a pathway to the source of their food. More pheromone is shed along the trail as more ants use it, which increases the possibility that more ants will follow it. The quickest route to the food attracts the most ants because they can journey there in the smallest amount of time.

It was in the well-known Double Bridge experiment that this phenomenon was first observed [47]. Ants were observed to choose the shortest path over a period of time when they were faced with a challenge of choosing a path to food source. The pheromone also evaporates over time in order to avoid solutions that are trapped in local minimum [48], thus reducing the likelihood that other ants will take the path. Also, as the speed at which pheromone is deposited exceeds its evaporation speed on the shortest path, the level of pheromone remains high. Consider the movement of ants between two nodes. Using pheromone deposits, the probability that an ant k located in node i will choose to go to another node j in the network is obtained in equation 1 below [48].

$$\mathrm{p}_{ij}^{k} = \begin{cases} \frac{(\tau_{ij}^{k})^{\alpha}(\eta_{ij}^{k})^{\beta}}{\sum_{i\in \mathrm{N}_i^k}(\tau_{ij}^{k})^{\alpha}(\eta_{ij}^{k})^{\beta}} & if\ j\ \in\ \mathrm{N}_i^k \\ 0 & if\ j\ \notin\ \mathrm{N}_i^k \end{cases} \quad (1)$$

Where τ_{ij}^{k} denotes pheromone levels. As for real ants, increased pheromone on a path attracts an ant consider that path more than other available pathways [49].

The heuristic function $\eta_{ij}^{k} = \frac{1}{d_{je}}$.

The pheromone update:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho * \Delta\tau_{ij}(t) + q * \Delta\tau_{ij}^{b}(t) \quad (2)$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad (3)$$

$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L_k}, if\ ant\ k\ passes\ node\ i\ and\ j \\ 0, \qquad\qquad otherwise \end{cases}$ is the quantity of pheromone deposited,

where Q is a constant and L_k is the total length of the path that ant k travels. A pseudocode of this algorithm is presented in Table A.2.

$$L_k = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (4)$$

A number of academicians have conducted extensive research on the operation mechanism, structural design, finding the best parameter values of ACO and proposed a number upgrades to solve these problems as seen in Table 3. Also, a number of variables listed in

Table 2 have been considered in their studies. The cross-path nodes produced throughout the path search process were optimized using ACO and geometric optimization by Liu et al. [50], allowing for enhanced algorithm's effectiveness and path quality through pheromone updates. You et al. [51] created a new heuristic operator to improve the population search's dissimilarity and convergence. Dai et al. [52] solved global convergence speed and path smoothing by improving the A* algorithm-based ACO algorithm and the maximum-minimum ant system. He also introduced a retraction mechanism to resolve the issue of the algorithm getting stuck at deadlocks. An adaptive state transfer and pheromone update method was proposed by Jiao et al. [53] to guarantee the importance of heuristic information and pheromone strength in the algorithm's iterative process, which improved the algorithm's adaptability for different environments to some extent, and ability to escape optimal

local solution. Khaled et al. [54] improved the state transfer formula to preferentially select the neighbour node with the most exits as the next node. This improved algorithm adds diversity to search process and suppresses the influence of invalid pheromones by dividing the multi heuristic function; rewarding and penalizing the worst path separately [55].

Table 2: Variables used in various ACO

Variables	Description
m	Ant's population size
N_{max}	Maximum iteration number
α	Weight of Pheromone
β	Weight of Heuristic information
ρ	Pheromone evaporation ratio
Q	Pheromone's Intensity
τ_{ij}	Pheromone on the path between i and j
η_{ij}	Heuristic information on j
ξ	Distance factor coefficient
φ	Distance correction parameter
υ	Ant's importance on moving straight
δ	Parameter to update Pheromone
γ	Diffusion coefficient

Table 3: Applied ant Colony Optimization Algorithm and its hybrids for path planning of smart vehicles

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
An improved ant colony algorithm (ACO-PD)	10	1.1	12	0.5		200		0.01	$\sqrt{2}$				Single	Static	Grid		• Proposed method solves convergence speed problem in ACO. • Geometric optimization method is implemented to improve path generated by ACO.	[50]
DL-ACO [PEACO and TPOA]	20 10	1 0.3	3 0.8	0.03 0.1		100 100	1	0.9				Roulette wheel selection	Single (Rikirobot)	Static	Grid		• PEACO and TPOA is combined to generate path and avoid obstacles.	[56]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	• Proposed method gives better results in path distance, smoothness and good solution rate when compared to APACA and MO-FA. • Experimentation is done with Rikirobot powered by Raspberry Pi and Rplidar A1. • Implements piecewise B-Spline to	

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	smoothen path.	
LF-ACO	50	1	3	0.3	100	50	1		10	5	20	Roulette wheel	Multiple robot	Static	Grid	MATLAB and Robotic Operating System (ROS)	• Proposed method aims to solve multi-robot path planning. • Pheromone update in ACO incorporates pheromones of leader and follower ant. • Maximum-minimum ant approach is employed for global search.	[55]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	• Generated path is optimized by TPOA and dynamic cut-point method.	
APACA	120	1	5	0.9	100	200							Single (Smart wheelchairs)	Static	20 x 20 Grid (see Figure 3a)		• Implementation of Direction determining Method to speed up convergence rate for global optimal search. • Proposed method shows better outcomes in	[53]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	number of iterations and path length when compared to IACA and GPACA.	
RMACA	50	1	5	0.5	10	54% lower than [57]							Single	Static	Grid. (Common, tunnel, trough, baffle maps)	MATLAB	• Retraction mechanism is employed to avoid local minimum. • Improved Maximum-minimum ant approach performs global search. • Estimation function in A*	[52]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	improves search efficiency of ACO. • RMACA is better in convergence rate and bending suppression effect.	
IACA	30	1	5		2	100							Single	Static	Grid	MATLAB	• Stimulating probability is introduced to improve transition rule. • Unlimited step length principle is used as heuristic information	[54]

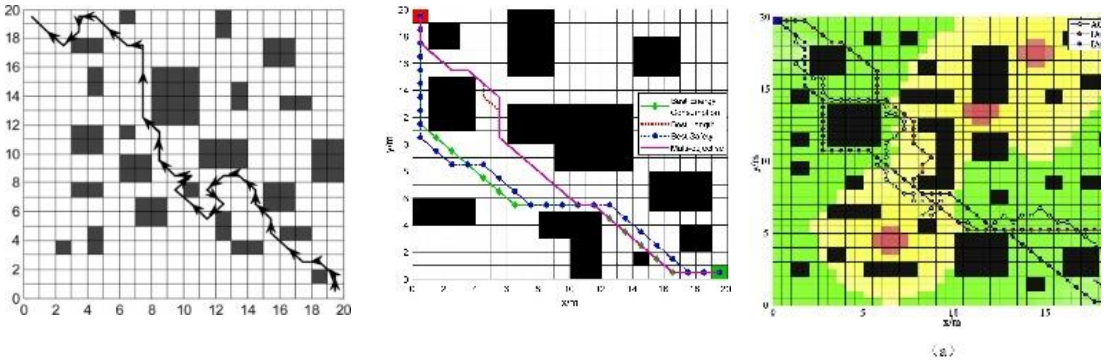
Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	for path search efficiency. • Dynamic change in evaporation rate increases convergence speed.	
Ant Colony Optimization and Fuzzy Control	80	1	9	0.5	1	100						Roulette wheel	Single	Static	20 x 20 Grid	MATLAB	• Fuzzy algorithm controls $\propto and\ \beta$ to adjust evaporation rate. • Proposed critical obstacle influence factor generates initial	[58]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	pheromone distribution.	
IACO-A*	50	0, 1, 2, 3, 4, 5	0, 1, 3, 5, 7, 9	0, 0.1, 0.3, 0.5, 0.7, 0.9	1	100							Single	Static	20 x 20 Grid (see Figure 3c)	MATLAB R2020b	• Proposed modelled environment is based on geometric modelling and Monte Carlo calculation. • Proposed method gives optimum results in terms of path length, cumulative radiation dose and energy consumption when	[59]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	compared to other algorithms.	
IAACO	50	1	7		2.5	100			1				Single	Static	20 x 20 Grid (see Figure 3b)		• The transition probability is induced with angle guiding factor and obstacle exclusion factor to enhance path search efficiency. • Heuristic information adaptive adjustment factor and adaptive pheromone volatilizatio	[60]

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of next node	Type of vehicle	Type of obstacles	Type of maps	Software	Remarks	Ref
																	n factor are introduced into the pheromone update rule for optimum global search.	

DL-ACO: Double Layer-ACO; PEACO: Parallel Elite Ant Colony Optimization; TPOA: Turning Point Optimization Algorithm; APACA: Adaptive Polymorphic Ant Colony Algorithm; Retraction Mechanism Ant Colony Algorithm; IACA: Improved Ant Colony Optimization Algorithm; MO-FA: Multiobjective Firefly Algorithm; IACO-A*: Improved Ant Colony Optimization algorithm-modified A*



a b c

Figure 3: Sample maps implemented for ant colony optimization algorithm: (a) APACA 20 x 20 grid map [53] (b) IAACO 20 x 20 grid map [60] (c) IACO-A* 20 x 20 grid map [59]

2.2.3 Particle Swarm Optimization (PSO)

The pursuit of food by a swarm of birds, a herd of tetrapods, or a school of fish serves as the basis for particle swarm optimization (PSO). Because there is no requirement for a leader to guide the group to the food source, this approach resembles the behaviour of these animals. [61]. The animals in this horde are unaware of the food's exact location, but they can estimate their distance from it. It is inefficient for each animal to try to get to the food source independently because it will take a long time and create pandemonium. As a result, the best strategy is to follow the members who are closest to the food source. [30]. Each individual animal represents the solution which contains this two information,

i. Their fitness value obtained from the objective function.

ii. The velocities directing the solution to the target location.

A set of solution or particle initializes the algorithm. Each particle searches the solutions space and returns their fitness value, p_{best}, in each iteration. Each iteration's best p_{best} value is saved as the global best value, g_{best}. Particle velocity and position are updated from these two values by the algorithm. The search is completed at the

maximum number of iterations or when the optimal solution is found. The formulas for updating the velocity and position of particles in PSO are as follows:

$$v_{id}(t+1) = w * v_{id}(t) + c_1 * r_1 * \left(p_{id} - x_{id}(t)\right) + c_2 * r_2 * \left(p_{gd} - x_{id}(t)\right) \quad (5)$$

$$x_{id}(t+1) = x_{id}(t) + \eta * v_{id}(t+1) \quad (6)$$

The inertia weight is denoted as w, c_1, c_2 are the learning factors, r_1, r_2 are normal distribution random numbers within the interval [0,1], η is denoted as the velocity constraint proportional factor, v_{id} is referred as the velocity of the i^{th} particle in d dimension, and x_{id} is referred as the position of the i^{th} particle in d dimension. A procedure to implement this algorithm is described in Table A.3.

This algorithm is comparable to the genetic algorithm in that both start with a set of populations that are formed randomly and evaluate the population according to its fitness value. This approach has been utilized for aerial robot navigation in unknown 3D environments [62], humanoid robot navigation [63] and in industrial robot navigation [64].

The effectiveness of PSO's performance is determined by how well its parameters are adjusted, controlled, and updated. A number of parameter modification and controlling procedures have been suggested since PSO was first introduced in 1995 in an effort to enhance its overall performance. Fix Inertia Weight (FIW) [65,66], Linearly Decreasing Inertia Weight (LDIW) [65–68], Time Varying Acceleration Coefficient (TVAC) [66,69], Random Inertia Weight (RANDIW) [65–67,69], Random Acceleration Coefficients (RANDAC) [70], and Fix Acceleration Coefficients (FAC) [65–67,69] are conventional parameter adjustment and controlling techniques. Various

studies outlined in Table 4 reveal proposed improvements and hybrids of PSO for solving path planning problem.

Harshal et. al [71] recommended an adaptive particle swarm optimization (APSO) that varies the inertia weight in every iteration. The algorithm explores the search space from a high inertia weight, avoiding local minimums during the planning process. As the number of iterations increase, the inertia weight decreases leading to an exploitation phase on the algorithm. This approach gave better results when compared to the basic PSO in terms of path length and planning time. The type of environment used for this study is shown in Figure 4a. In enhancing sampling of points in probabilistic roadmap method (PRM), Qisen et. al [72] proposed PSO-PRM hybrid approach. This method gains knowledge of sample points on obstacle regions and uses it to improve sample points in free space, especially at narrow passages, thus improving connectivity in those areas. Ellips et. al [73] employed PSO to achieve shortest and smooth feasible path. Particles are initialized based on the number of points formed in free space from a robot laser sensor. The best particle is selected and its new position is obtained based on the robot's sensor. Probabilistic roadmap method (PRM) is used as a local planner for obstacle avoidance. Simulations results shows a better runtime of the proposed method when compared with the basic PRM approach.

Xun et. al [74] brought forward an improved PSO (IPSO) which initializes particles through uniform random distribution, implements inertia weight of exponential decay to enhance planning time, improves the social and cognitive factors and smoothens path using cubic spline interpolation. This approach was tested on benchmark functions with some other proposed PSO variants and the results were compared based on some performance index (average value, best result, standard deviation, running

time). Also comparison was made based on shortest path and average planning time with various PSO variants and IPSO showed better results. Baoye et. al [75] developed a PSO variant (FOPSO) that introduces an adaptive fractional-order velocity and utilizes Bezier curve to smoothen planned path. This variant was tested on benchmark functions to analyse its performance in comparison to some PSO variants. Shahab et. al [23] implemented PSO in sampling points randomly along grid lines of between start and goal points. First, points are spaced out and placed along Euclidean path from start to goal points, the PSO samples points along the grid lines of initially spaced out points. To assess the efficacy of this proposed approach, simulations were run in various environments with static obstacles.

Table 4: Applied particle Swarm Optimization and its hybrids for path planning of smart vehicles

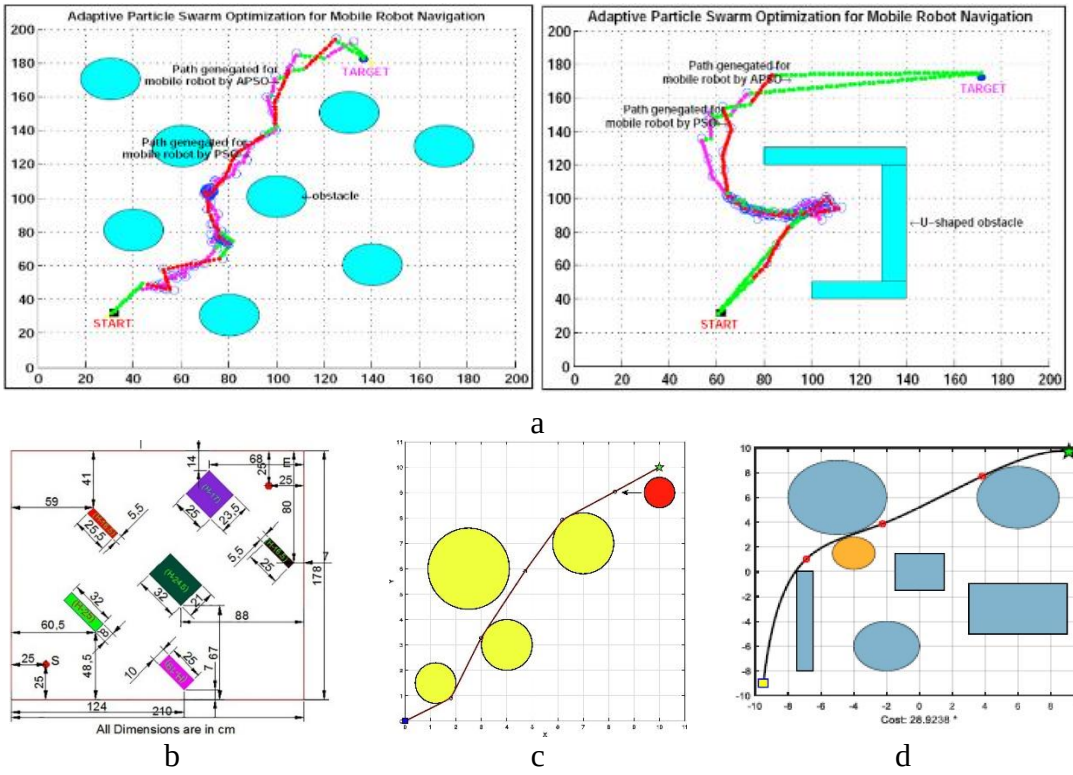
Types	Particle size	Inertia weight (w)	Cognitive fact or (c1)	Social factor (c2)	Number of generations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
Hybrid PSO	20	10%	1.5	1.5	500	Single (Mobile robot)	Dynamic	200 x 200 Geometrical	MATLAB 2018b	• The performance of hybrid PSO and ACO on shortest path and least time constraint is measured against PSO and ACO separately. • Although it's observed that PSO outperforms ACO, the hybrid gives a superior results.	[76]
EDPSO	150	1	0.4	0.4	150	Single	Static and Dynamic	20 x 20 Geometrical (see Figure 4d)		• Peaks of diversity in population gives room for more	[77]

Types	Particle size	Inertia weight (w)	Cognitive fact or (c1)	Social factor (c2)	Number of generations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
										exploration in the search space. • Can be applied to smart vehicles with slow movements. • Comparison was made on all the cited meta-heuristics using 10 complex multi-model functions from the CEC 2019 benchmarking suite.	
PSO-AWDV	200	0.9, 0.5	2.0, 1.0	1.0, 2.0	100	Single	Static	11x 11 Geometrical (see Figure 4c)		• Quartic Bezier transition curve with three control points is implemented	[78]

Types	Particle size	Inertia weight (w)	Cognitive fact or (c1)	Social factor (c2)	Number of generations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
										to smoothen planned path.	
Improved PSO	10	0.4, 0.9	2	2	3000	Single	Static	18 x 18 Geometrical	MATLAB R2016a	• Proposed mutation operation increases particle diversity. • Proposed de-redundant algorithm removes needless points, thus improving generated path.	[79]
FIMOPSO	50	0.4 to 0.9			100	Single	Static and dynamic	210 x 178 Geometrical (see Figure 4b)	MATLAB	• Constraints to be minimized are path length, motor torque, travel time, robot acceleration; obstacle	[80]

Types	Particle size	Inertia weight (w)	Cognitive fact or (c1)	Social factor (c2)	Number of generations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
										avoidance is maximized. • Obstacle avoidance problem is solved with Fuzzy inference system.	

EDPSO: Enhanced Diversity Particle Swarm Optimization; PSO-AWDV: Particle Swarm Optimization - Adaptive Weighted Delay Velocity; FIMOPSO: Fuzzy enhanced Improved Multi-Objective Particle Swarm Optimization



a b c d

Figure 4: Sample maps implemented for particle swarm optimization : (a) APSO 200 x 200 map [71] (b) FIMOPSO 210 x 178 map [80] (c) PSO-AWDV 11 x 11 map [78] (d) EDPSO 20 x 20 map [77]

2.2.4 Artificial Bee Colony (ABC)

Artificial bee colony algorithm, proposed by Dervis Karaboga to improve the polynomial mathematical issues [81], is a technique centred on the hunting behaviour of honey bees for food. Honey bees utilize pheromone and waggle dance to communicate. A bee that discovers a food supply assesses the amount of nectar, goes back to the hive, and performs a waggle dance to exchange the food source. The nectar quantity in the food source is determined by the intensity of the waggle dance. The position of a food supply for an optimization problem indicates one potential solution, and its fitness is reflected in the nectar content of the food source.

The bees are separated into three groups by the ABC algorithm: employed workers, observers, and scout bees. For each food source position, it is assumed that there is just

one artificially hired bee. Through the waggle dance, employed bees can signal the location of food supplies to observer bees. The watchful bees choose food sources based on their quality. This implies that sources of food with higher quality are more likely to be chosen. Employed bees change into scout bees to look for new food sources when they leave a food source position. Scout bees memorize the quality of food spots while searching and contrast them with ones already discovered to determine the best one. The pseudocode for this algorithm can be found in Table A.4.

At the beginning, a population of food source position (SN) is determined (SN is the size of the population), where each food source/solution is a D-dimensional vector (D is the number of optimization parameters). Each food source is associated with a probability p_i which affects the choices of observer bees.

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{7}$$

where fit_i is the fitness of solution i, and SN is number of employed bees (population size). New food position from old one is calculated from the expression below:

$$v_{ij} = x_{ij} + \emptyset_{ij}(x_{ij} - x_{kj}) \tag{8}$$

where k and j are random values in the set {1, 2,…,SN} and {1,2,…,D} respectively. k should be different from i. $\emptyset_{ij} \in [-1,1]$ controls the production of a neighbour food source position around x_{ij}.

Research on ABC for path planning has led to various improvements and hybrids of ABC as outlined in Table 5. In navigating smart vehicles, [82] proposed a combine ABC and evolutionary programming (EP) approach. The ABC algorithm was used as the local search method while the EP was implemented to enhance the obtained possible path. This work was implemented with multi-robot and improved for an

unfamiliar environment with dynamic obstacles [83]. Despite the improvement, [84] observed some flaws such as the distance of new bee position and obstacle not being considered, and proposed an improved ABC-EP approach. Adaptive Dimension Limit-Artificial Bee Colony Algorithm (ADL-ABC) was suggested by [85] to determine optimal global path for mobile robot. Results show that it could find the solution with few iterations and less computational time. A developed Directed Artificial Bee Colony achieved better results in a dense environment (i.e. an environment with many static obstacles) with comparison to other state-of-the-art algorithms [86]. To limit the computational time effort in real-time path planning, [87] proposed a combined Artificial Bee Colony and Dijkstra algorithm.

Table 5: Artificial Bee Colony Algorithm for path planning of smart vehicles

Types	Population size	Number of iterations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
ADL-ABC	100	1000	Single	Static	10 x 10 Geometrical	MATLAB	• Implemented dynamic control limit reduces computational time and number of iterations. • Generated path is smoothened using cubic polynomial through three via points. • Better results are observed when proposed method is compared to ABC.	[85]
ABC-EP	10	500 generations (for EP)	Single (Xidoo-Bot, mobile robot Pioneer 3-AT)	Static	Various Geometrical and Grid maps. (see Figure 5)	C language	• While ABC builds a path in collision free space, EP improves the path using mutation to produce a short path. • Proposed approach was implemented in some benchmark maps. • ABC-EP is deployed to an experimental platform to show its feasibility.	[82]

Types	Population size	Number of iterations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
Improved ABC-EP	200, 400, 600, 800, 1000	100	Single	Static and Dynamic	100m x 100m Geometric	MATLAB 2018b	• Best food points among randomly distributed ones which aid in finding optimum path are selected based on its distance to goal point and nearest obstacle. • When determining the optimum path, proposed algorithm takes into account the distance between the new bee position (best node) and any surrounding barriers.	[84]
Enhanced ABC	25	100	Single	Static	100 x 100 Grid		• Cubic Ferguson spline is introduced to smoothen path generated by ABC.	[88]

ADL-ABC: Adaptive Dimension Limit – Artificial Bee Colony; ABC-EP: Artificial Bee Colony – Evolutionary Programming;

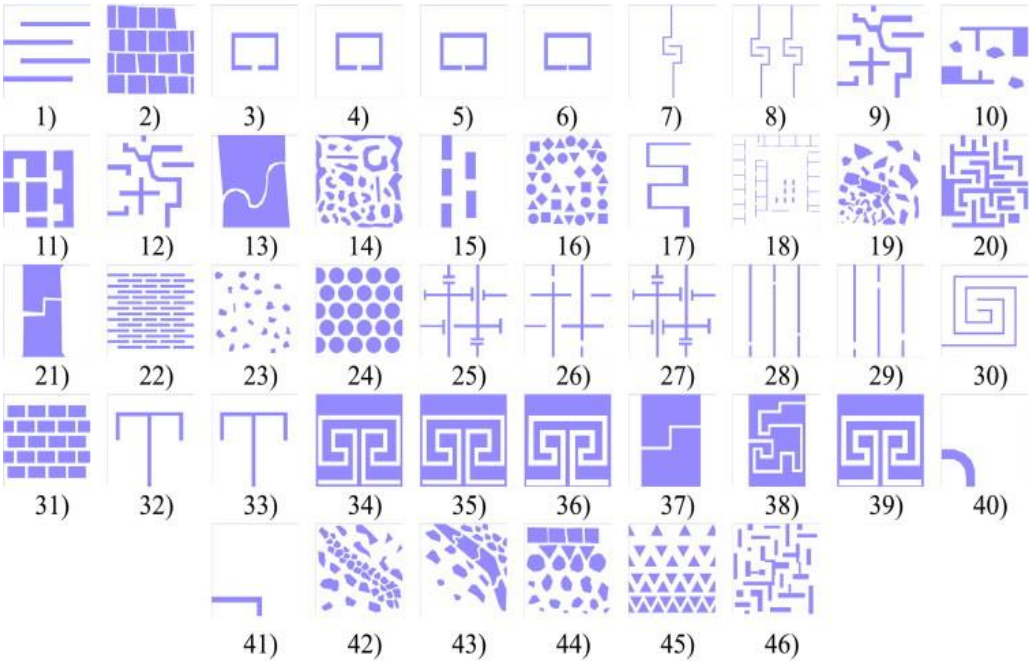


Figure 5: List of benchmark maps used in ABC-EP [82]

2.2.5 Firefly Algorithm (FA)

Firefly algorithm developed by [89], was stirred from the flashing behaviour of fireflies. They communicate among themselves through their specific light patterns. How bright the light of a firefly shines equates to its attractiveness. This attractiveness is regarded as the fitness value in FFA. For any two fireflies, the brighter firefly attracts dim one. The attractiveness of a firefly can be represented in the equation below [90].

$$\beta = \beta_0 e^{-\gamma r_{ij}^2} \tag{9}$$

where β represents the attractiveness, $\beta_0$the maximum attractiveness value which can be 1 in most situations, γ the light absorption coefficient which lies between [0.1, 10], and r_{ij} the distance between firefly i and firefly j which is calculated using the standard Euclidean distance.

$$r_{ij} = x_i + \beta(x_j - x_i) + \alpha(\phi - 0.5) \tag{10}$$

Where x_i and x_j are the positions of firefly i and firefly j, α and ϕ are random numbers in the distribution [0, 1]. The pseudocode for firefly algorithm is described in Table A.5.

Firefly algorithm implementation in various research work can be seen in robotics [91,92], machine learning [93], journalism [94], cloud computing [95]. A list of proposed techniques for its implementation for smart vehicles is revealed in Table 6. In improving convergence speed and local search accuracy of standard firefly algorithm, [96] proposed a modified firefly algorithm (PPMFA) that infuses a Gaussian random number to the fixed step size. The randomized value improves the diversity to firefly population so as to prevent the algorithm from getting stuck at dead end zones. Also a path centre technique was introduced to calculate two fireflies distance from each other (i.e. two paths). At first, the geometric centres of each segment in a path are connected to form new segments. The midpoints of these new segments are connected and the process is repeated for new segments until one segment is left which is referred to as path centre. The distance between to path centres is assumed to be the distance between two fireflies. This modified firefly algorithm gave better results in terms of accuracy and convergence speed when compared to particle swarm optimization (PSO) and standard firefly algorithm (SFA). [97] suggested Developed firefly algorithm (DFA) to solve multi-objective navigation problem. Grid map extension was done on the utilized map in order to have feasible paths. Comparison and segregation of paths, also known as fireflies is done using Pareto dominance relationship. Non-dominant fireflies are kept in a created elite record library and compared to other fireflies during iteration process. This algorithm also involves an evolutionary stage which adds, removes or swaps points on a planning

path to optimize it. DFA displayed better efficiency when tested with NSGA-II on a ZDT1 instance.

[98] applied firefly algorithm to navigate mobile robot in a map with dynamic obstacles. Artificial intelligent mechanism is implemented to navigate the robot from start to goal point while a controller which implements firefly algorithm is developed to detect and avoid obstacles. The controller generates fireflies (or paths) when the robot approaches an obstacle and the brightest firefly (i.e. the safest and optimum path from the obstacle) is selected through Euclidean distance between it and the closest obstacle. The execution of this approach outweighs other intelligent approaches when compared in terms of path length in three different scenarios. Experimentation with Khepera-II robot was carried out and its deviation from simulated results was about 5.7%. Alia et. al [99] developed a modified version of firefly algorithm to plan path of mobile robot in a 3D sphere partially dynamic environment. A firefly is considered an agent that navigates to a point around an obstacle. The generated paths are regarded as possible solutions, of which the best path is chosen based on path length and the completeness of the path. It is observed that this modified approach requires no large memory space during implementation and works smoothly in sphere space.

Table 6: Firefly Algorithm for path planning of smart vehicles

Type	Population size	Generation	γ	β_0	α	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
FFA						Single	Static	10 x 10 Geometrical		• A* algorithm is used to find the shortest path. • Cubic polynomial spline is interpolated on the generated path to produce smooth trajectory using iterative random selection.	[91]
FAMCPSO			1	2	0.5	Single	Static	600cm x 800cm Geometrical	MATLAB 2018b	• Proposed method is a combination of MCPSO and FA. • Consideration of inverse dynamic and	[100]

Type	Population size	Generation	γ	β_0	α	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
										kinematic modelling to obtain optimum torque and velocity for wheels of AMR. • The recommended hybrid method shows good results when compared to various algorithms in different environments.	
MO-FA	200	150				Single	Static	Grid (see Figure 6a)	C/C++ language	• Path safety, length, and smoothness are considered in the design of	[101]

Type	Population size	Generation	γ	β_0	α	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
										proposed method.	
FA-TPM	5 - 100	50 – 100	0.1 - 1	0.1 - 1	0.1 - 1	Single (Fire Bird V robot-NEX Robotics and Embedded Real-Time Systems Lab, CSE IIT Bombay)	Dynamic	Grid (see Figure 6b)	Microsoft Visual C++, 2010 with OpenGL	• While TPM searches for obstacle free path, FA performs obstacle avoidance.	[102]

FAMCPSO: Firefly Algorithm Modified Chaotic Particle Swarm Optimization; AMR: Autonomous Mobile robot; MCPSO: Modified Chaotic Particle Swarm Optimization; FA-TPM: Firefly Algorithm-Three Path Method

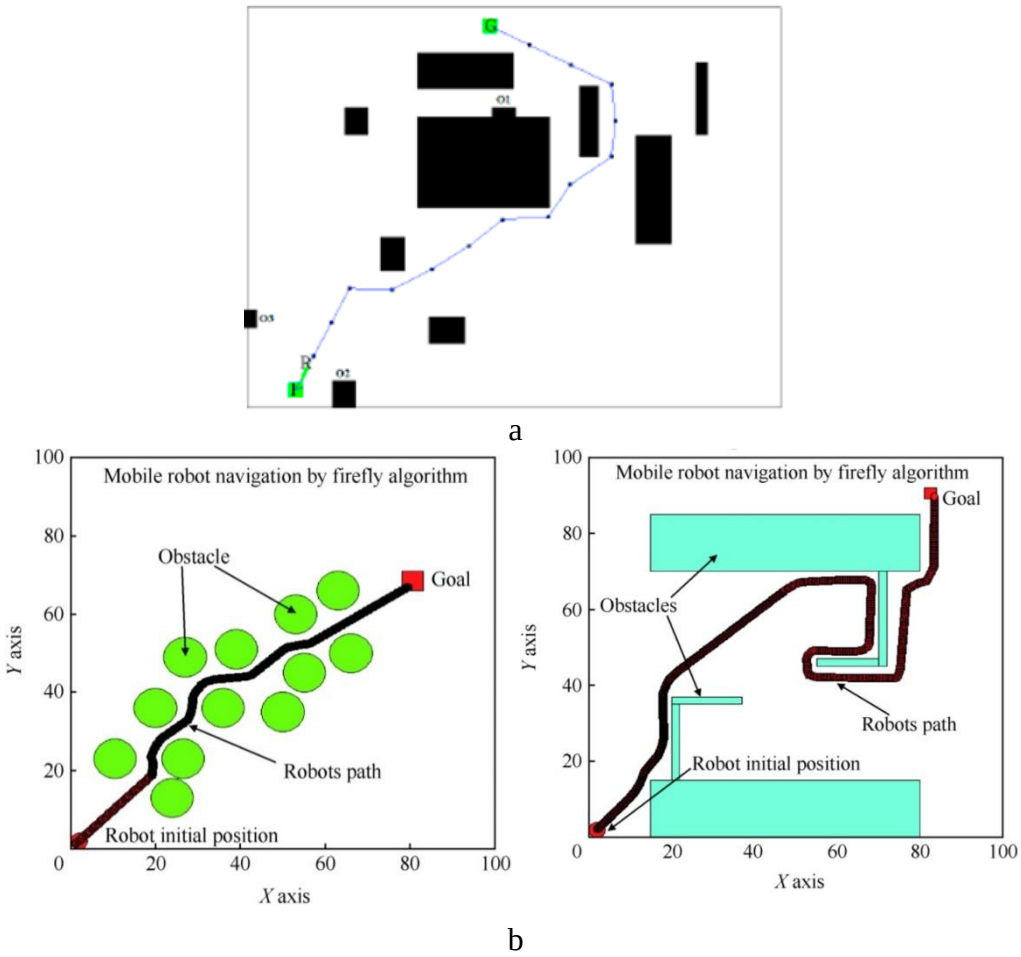


Figure 6: Sample maps implemented for firefly algorithm : (a) FA-TPM [102] (b) FA 100 x 100 map [98]

2.2.6 Cuckoo Search Algorithm (CSA)

Cuckoo search, a nature-inspired algorithm was developed in 2009 to solve optimization problems [103]. It is centred on brood parasitism of cuckoo species in their breeding and laying of eggs. In the search for host bird's nest to lay their eggs, they can go as far as emulating the colour and pattern of egg laying of host species [104]. This fiddle and shading tricks give a very low probability for the host species to identity the odd egg in its nest [25]. Peradventure the host specie discovers the odd egg, it evicts the eggs from its nest or leaves to build a new habitation. Cuckoo eggs hatch faster than host eggs, allowing it to evict the host's eggs, increasing the cuckoo

chick's portion of food provided by the host [105]. In the behavioural rules of cuckoo birds:

- Every cuckoo selects a nest at random to place an egg in turn.
- The nest that contains the cuckoo which survives being evicted by the host specie will be moved to the next generation.
- A host has the probability $P \in [0,1]$ to discover the odd egg and the total number of accessible eggs within the search space is fixed.

The search for a nest is done by random walk, although levy flight [103] is the most widely used search method. To simply solve path planning problems, host eggs in a nest can represent solutions and a cuckoo egg in the nest a new solution. This aims at replacing the poor performing solutions with better solution (cuckoo's egg). Implementing levy flight search for a new solution x^{t+1} is done with this mathematical model.

$$x_i{}^{t+1} = x_i{}^{t} + \alpha \oplus Levy(\gamma) \tag{11}$$

where i represents the ith particle, t stands for the iteration cycle, $\alpha > 0$ is the step size, and $\oplus$ means entry-wise multiplication. Step lengths of Levy flight are distributed according to this probability.

$$Levy(\gamma) = L^{-\gamma}, (1 < \gamma \leq 3) \tag{12}$$

where L represents the length of the step size and γ denotes the variance. P, γ, and α are key parameters that need to be fine-tuned to find improved solutions. Less fine-tuning parameters is one of the pros of cuckoo search together with its possibility to deal with multi-modal objective functions. A pidgin code to implement this algorithm is described in Table A.6.

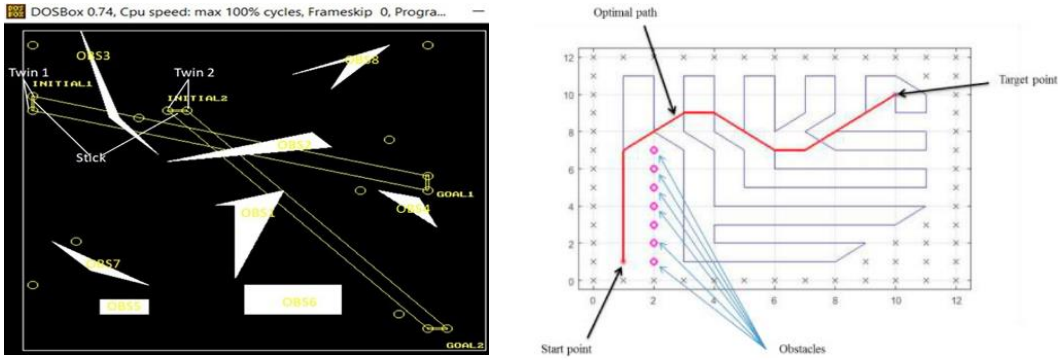
Research carried out in implementing cuckoo search in various fields range from its implementation in vehicle routing problem [106,107], neural networks [108], scheduling [109,110], medical [111,112], cloud computing [113] to robotics, especially navigation of smart vehicle. A variety of hybrids of Cuckoo Search for path planning problem have been suggested as seen in Table 7. To solve multi-robot collaboration and navigation in a dense obstacle map, [114] proposed Modified Cuckoo Search as an innovative approach. [22] compared cuckoo search and other metaheuristic algorithms with classical path planning algorithms in different path planning scenarios. A hybrid CSA with firefly and PSO was proposed to minimize computational cost and maximize algorithm efficiency for path planning of mobile robots [115]. [92] used cuckoo search to prevent the proposed quantum firefly algorithm from premature convergence. In order to optimize mobile robot path in an environment with changing obstacles, [116] implemented a modified cuckoo search algorithm to process the obstacle distance and heading angle information coming from the robot sensor.

Table 7: Cuckoo Search Algorithm for path planning of smart vehicles

Type	P	γ	α	Population size	Number generations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
MCS-SCA-PSO	0.25			30		Multiple (Epuck robot)	Static and dynamic	450 x 450 Geometrical (see Figure 7a)	C language	• PSO performs local search, CSA performs global search, and sine cosine algorithm implements greedy approach.	[117]
Improved CSA	0.25			30		Single	Dynamic	Topological	MATLAB 2014a	• Global search ability is improved by introducing mutation and crossover. • Convergence rate and optimization accuracy of algorithm are tested using unimodal and multimodal functions.	[118]
Hybrid CSA-BA				20	500	Single	Static	12 x 12 Geometrical (see Figure 7b)	MATLAB	• The proposed approach is implemented in two environments with various positions of circular obstacles.	[119]

Type	P	γ	α	Population size	Number generations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
Hybrid genetic-cucking	0.25		1	400	40	Single	Static	10000 x 8000 x 5000 Geometrical	MATLAB	• Using intelligent algorithms in path planning of 3D environment is studied. • Spherical obstacles are implemented.	[120]
CSA-PSO-FA		1	0.2	20	1000	Single and multiple (Kobuki mobile robot)	Static	200 x 160 and 100 x 100 Grid	MATLAB, ROS	• CS-PSO-FA algorithm is investigated both in simulation and experimentally.	[115]
CSA-BA	0 - 1			30	500	Single	Static	12 x 12 Geometrical	MATLAB 2015b	• CSA-BA obtains a better result in path length than CSA and BA separately.	[121]

CSA-BA: Cuckoo Search Algorithm – Bat Algorithm; CSA-PSO-FA: Cuckoo Search Algorithm – Particle Swarm Optimization – Firefly Algorithm; MCS-SCA-PSO: Modified Cuckoo Search - Sine Cosine Algorithm - Particle Swarm Optimization



a b

Figure 7: Sample maps implemented for cuckoo search algorithm: (a) MCS-SCA-PSO 450 x 450 map [117] (b) Hybrid CSA-BA 12 x 12 map [119]

2.2.7 Whale Optimization Algorithm (WOA)

Mirjalili et. al [122] proposed the whale optimization algorithm which simulates the bubble-net hunting strategy of humpback whales [123]. Humpback whales mostly live in groups and feed on krill and small fish herds close to the water body surface. During foraging a humpback whale can dive around 12m deep into the water, then begin to create distinct bubbles around a prey while it swims spirally upward towards the surface. Its prey encircling, spiral bubble-net manoeuvre and prey search can be modelled mathematically.

- Encircling prey

  A humpback whale encircles a prey when it recognizes its location. WOA assumes the best solution is the whale (search agent) that's close to the target prey. Other search agents adjust their position in relation to the best agent. This behaviour is seen in the following equation:

$$D = |C.X^*(t) - X(t)| \quad (13)$$

$$X(t+1) = X^*(t) - A.D \quad (14)$$

$$A = 2a.r - a \quad (15)$$

$$C = 2r \quad (16)$$

where A and C are coefficients, t denotes current iteration, X^* is the best whale position, X is the current whale position, a is a decreasing constant from 2 to 0 which is expressed as $a = 2 - \frac{2t}{M}$ (M: maximum number of iteration), $r \in [0,1]$ is a random value.

- Spiral bubble-net manoeuvre (Exploitation phase)

  As the value of a decreases, the encircling of prey shrinks, given A to be a random vale in [-a, a], search agents can obtain the relationship between their position and current best position when A is reduced to [-1, 1]. Furthermore, during the spiral movement, the whale's position to the prey is updated and the distance D^' between the ith whale and the prey (best solution obtained so far) is calculated as follows:

$$X(t+1) = D'.e^{bl}.\cos(2\pi l) + X^*(t) \quad (17)$$

$$D' = |X^*(t) - X(t)| \quad (18)$$

  where b is a constant for defining logarithmic spiral shape, $l \in [-1,1]$ is a random value. During whale hunting, the encircling of prey shrinks while whale moves in spiral motion at the same time, therefore the whales position is updated while it's encircling or while it's in spiral movement on a probability of 50% as seen in the equation below.

$$x(t+1) = \begin{cases} X^*(t) - A.D, & p < 0.5 \\ D'.e^{bl}.\cos(2\pi l) + X^*(t), & p \geq 0.5 \end{cases} \quad (19)$$

- Searching for prey (Exploration phase)

  During the search phase, a whale's position is updated according to a random agent i.e. when $|A| > 1$, not the current best agent i.e. when $|A| < 1$. A global search is performed when $|A| > 1$. Given X_{rand} to be a random agent's position, a whales updated position is given as:

$$D = |C.X_{rand} - X| \quad (20)$$

$$X(t+1) = X_{rand} - A.D \tag{21}$$

A pseudocode of this algorithm is described in Table A.7. Researchers have applied whale optimization algorithm for image segmentation [124], validation of welded Al/Cu bimetal sheet [125], intelligent facial emotion recognition [126], improve power system stabilizer [127], task scheduling in microprocessor system [128].

In the field of robotics, it has been implemented to plan joint trajectory of robotic arm [129],for robotic manufacturing [130], to plan navigation of unmanned vehicles [131], for multiple robot space exploration [132,133], for deep neural networks [134]. Table 8 lists various studies of its implementation in smart vehicle navigation.

Table 8: Whale optimization algorithm for path planning of smart vehicles

Type	Population size	Number of iterations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
WOA		100	Single (Khepera II mobile robot)	Static	Geometrical	MATLAB	• Algorithm solves mobile robot scheduling problem in a manufacturing environment. • Novel mathematical model is proposed and experimentally tested on 26 benchmark functions.	[135]
Improved WOA based on GA		100	Single	Static	20 x 20 Grid		• Proposed method can be implemented for logistic mobile robot. • Efficiency of proposed algorithm is improved by 10.71% compared to traditional WOA.	[136]
MWOA	100	500	Single	Static	300 x 500 pixels Geometrical		• Distance and smooth path functions are minimized. • The pareto front-optimal solution gives the optimal solution for MWOA. • Proposed method has a lower error rate than the Multi-Objective Genetic Algorithm (MOGA) method [137].	[138]

Type	Population size	Number of iterations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
MO-WOA	50, 80, 100, 150	50, 70, 90, 110	Multiple	Static	15 x 15 Grid (see Figure 8)	MATLAB	• At 130 iterations and 150 way-points, proposed algorithm outperforms compared deterministic and hybrid stochastic exploration algorithm. • Map exploration and minimum time map enhancing accuracy is the idea behind proposed algorithm.	[139]
NWOA		1000	Single (Raspberry Pi (3B+))	Static and dynamic	1800 x 1800 Geometrical	Python	• Adaptive technology, enhanced potential field factors and virtual obstacles are introduced to optimize the convergence rate of the algorithm. • NWOA performance better in convergence rate when compared to WOA, GA-WOA, and EGE-WOA.	[140]
Updated WOA		500	Single	Static	8 x 8 Geometrical		• Proposes a changed whale advancement calculation based Mobile robot way determination.	[141]

MWOA: Multiobjective Whale Optimization Algorithm; MO-WOA: Multi-Objective Whale Optimization Algorithm; NWOA: Novel Whale Optimization Algorithm.

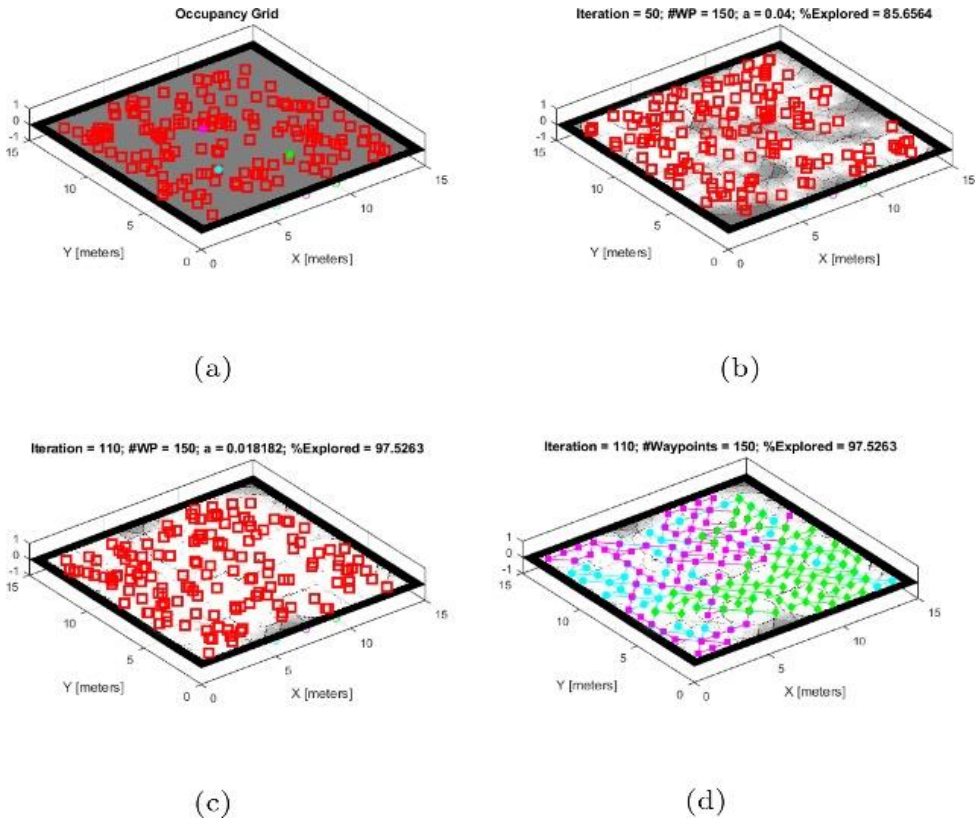


Figure 8: Sample map implemented for whale optimization algorithm: MO-WOA 15 x 15 map [139]

2.2.8 Grey Wolf Optimization (GWO)

Grey wolf optimization, proposed by [142] [142] simulates the natural leadership pyramid and hunting technique of grey wolves. Grey wolves belong to set of apex predators located at the top of the food chain order. They prefer to live in groups (packs) of 5 – 12 on average and each individual in the group possess strict social hierarchy as demonstrated in Figure 9.

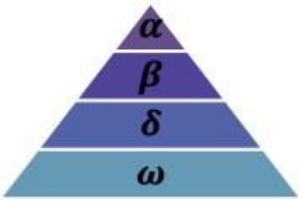


Figure 9: Social hierarchy of grey wolves

The alphas (α) are the leaders, a male and a female. Being at the top of the hierarchy, they are considered the fittest solution and every order given by them must be obeyed by other wolves in the group. The beta (β) wolves are ranked second and are considered the best candidate to be alpha. The delta (δ) wolves which constitute the scouts, sentinels, elders, hunters, caretakers dominate the omega (ω) wolves which are considered as scapegoats in the park. In addition to this social hierarchy, grey wolves hunt their prey in their packs. Starting from tracking, chasing and approaching the prey, they pursue, encompass and harass their prey until it becomes too weak to resist the pack, then it is attacked. This behaviour can be modelled mathematically by assuming the fittest solution as the alpha wolf, the beta wolf being the second best solution, the delta wolf as the third best and the rest of the solutions as the omega wolf. The mathematical model is considered under these measures:

- Encircling the prey:

$$\vec{D} = \left|\vec{C}\,\overrightarrow{X_P}(t) - \overrightarrow{X_P}(t)\right| \tag{22}$$

$$\vec{X}(t+1) = \overrightarrow{X_P}(t) - \vec{A}\,\vec{D} \tag{23}$$

$$\vec{A} = 2\vec{a}\,\overrightarrow{r_1} - \vec{a} \tag{24}$$

$$\vec{C} = 2\overrightarrow{r_2} \tag{25}$$

  where $\vec{A}$ and $\vec{C}$ are coefficient vectors, t is the current iteration, $\overrightarrow{X_P}$ is the position vector of the prey, $\vec{X}$ is the position vector of the grey wolf, $\overrightarrow{r_1}$,$\overrightarrow{r_2}$ are random vectors in [0,1], and $\vec{a}$ is decreased linearly from 2 to 0 over the course of iteration.

- Hunting:

  Because it is presumed that they have more knowledge about the location of prospective prey, alpha, beta, and delta wolves update their positions before omega wolves do for each iteration. Their position update is indicated in this formulas:

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}\,\overrightarrow{X_\alpha}(t) - \vec{X}(t)\right| \quad (26)$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2}\,\overrightarrow{X_\beta}(t) - \vec{X}(t)\right| \quad (27)$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3}\,\overrightarrow{X_\delta}(t) - \vec{X}(t)\right| \quad (28)$$

$$\overrightarrow{X_1}(t+1) = \overrightarrow{X_\alpha}(t) - \overrightarrow{A_1}\,\overrightarrow{D_\alpha} \quad (29)$$

$$\overrightarrow{X_2}(t+1) = \overrightarrow{X_\beta}(t) - \overrightarrow{A_2}\,\overrightarrow{D_\beta} \quad (30)$$

$$\overrightarrow{X_3}(t+1) = \overrightarrow{X_\delta}(t) - \overrightarrow{A_3}\,\overrightarrow{D_\delta} \quad (31)$$

$$\vec{X}(t+1) = (\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3})/3 \quad (32)$$

- Attacking Prey (Exploitation):

  The prey attack can be mathematically represented by decreasing the value of $\vec{a}$ from 2 to 0 over the course of iteration. The grey wolf attacks the prey when $|A| < 1$ while $\vec{A}$ is a random value in [-2$\vec{a}$, 2$\vec{a}$].

- Searching for prey (Exploration):

  The wolves are forced to explore for a fitter prey when$|A| > 1$. A parameter $\vec{C}$ is a random value in [0, 2] which assist GWO to exhibit a random behaviour by stochastically emphasizing (C > 1) or deemphasizing (C < 1) the attack of grey wolves. Pseudocode of grey wolf optimization is shown in Table A.8.

GWO has been implemented to solve optimization problems in various fields such as in medicine [143,144], operation sequencing [145], unmanned aerial vehicle [146], multi agent systems [147], robotics (see a variety of proposed GWO for smart vehicle navigation in Table 9). [148] proposed a hybrid PSO-GWO algorithm to optimize the path length and provide a smooth trajectory for mobile robots. He then introduced mutation operator to smoothen the trajectory generated by his proposed PSO-GWO algorithm [149]. A modified position update mechanism was suggested by [150] to

solve local minimum of GWO for robot path planning. [151] recommended a Variable Weight GWO to enhance speed and shorten planned distance of mobile robot.

Table 9: Grey Wolf Optimization for path planning of smart vehicles

Type	Population size	Iterations	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
IGWO	30	100	Single	Static	Geometrical	MATLAB R2018b	• The algorithm is tested on 20 benchmark functions.	[150]
VM-GWO	20,25 (For map 1) 25,30 (For map2)	35,20 (For map 1) 30,40(For map 2)	Single	Static	Geometrical (3D map)	MATLAB 2018a	• Execution speed outperformed GWO.	[151]
HPSO-GWO-EA			Single	Static	Geometrical	MATLAB R2017a	• Mutation operator from evolution algorithm is introduced to solve path length and smoothness.	[148]
HPSO-GWO	100	500	Single	Static and Dynamic	100 x 100 Geometrical (see Figure 10)	MATLAB R2019b	• Frequency-based function is introduced to modify the search process of GWO.	[149]
HWGO	20	100	Single			MATLAB R2019	• Proposed algorithm is implemented in tuning parameters of fractional order PID controller.	[152]

IGWO: Improved Grey Wolf Optimization; VM-GWO: Variable Weight - Grey Wolf Optimization; HPSO-GWO-EA: Hybrid Particle Swarm Optimization - Grey Wolf Optimization – Evolution Algorithm; HPSO-GWO: Hybrid Particle Swarm Optimization - Grey Wolf Optimization; HWGO: Hybrid Whale Grey Wolf Optimizer;

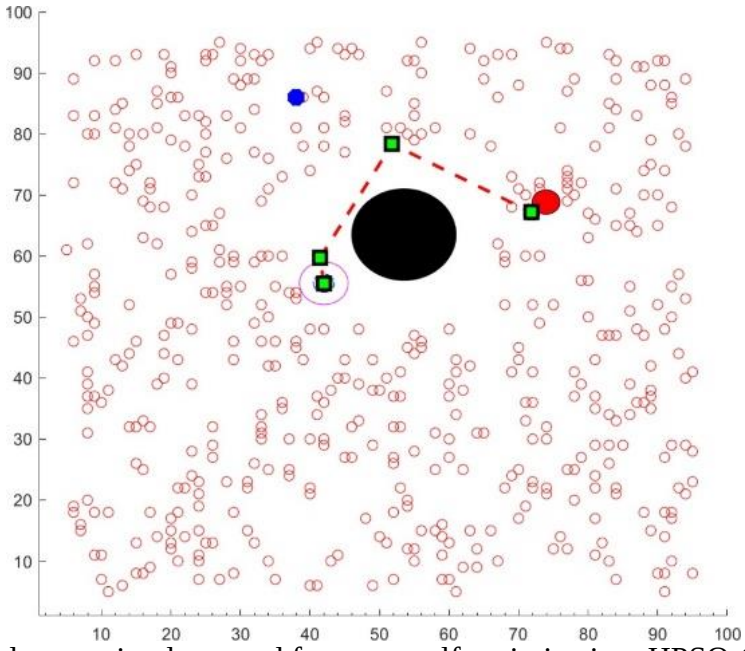


Figure 10: Sample maps implemented for grey wolf optimization: HPSO-GWO 100 x 100 [149]

2.2.9 Multi-Verse Optimizer (MVO)

Designed based on inspiration from the theory of multi-verse in physics, it is a population-based algorithm built on three cosmological concepts: white hole, black hole and worm hole [153]. In contrast to universe, multi-verse refers to the existence of other universes which interact together in the multi-verse theory [154]. Big bang [155], which is probably the key factor for the birth of the universe is considered as a white hole by physicists [156]. Black holes, whose behaviour is dissimilar to white holes attract light beams with high gravitational force [157]. Worm holes which act as time travel tunnel connect different parts of a universe or even connects universes. Universe expand through space due to a factor called inflation rate (eternal inflation) [158]. Mathematical models have been built on these three concepts to carry out exploration, exploitation and local search respectively.

MVO employs white and black holes to explore search spaces while worm hole exploits search spaces. The universe is equated to a solution and objects in the universe

is analogous to variables. The inflation rate of a solution is the fitness value of that solution. Universes of MVO follow these rules:

1. High inflation rate leads to a high chance of having a white hole.
2. Low inflation rate leads to a low chance of having a black hole.
3. High inflation rate universes are likely to pass objects through white holes.
4. Lower inflation rate universes have a tendency to get objects through black holes.
5. Regardless of inflation rate, objects in all universes can move randomly towards an optimal universe via wormholes.

Assume that U is a set of universes, where n is the number of possible solutions (universes) and d represents the number of parameters or variables:

$$U = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \tag{33}$$

then each parameter can be represented as below:

$$x_i^j = \begin{cases} x_k^j, & r1 < NI(Ui) \\ x_i^j, & r1 \geq NI(Ui) \end{cases} \tag{34}$$

x_i^j is the jth parameter of the ith universe. $NI(Ui)$ indicates the normalized inflation rate of the ith universe. Ui is the ith universe. $r1$ is a random value in [0, 1]. x_k^j denotes the jth variable of kth universe chosen by a roulette wheel selection mechanism. Roulette wheel, which depends on normalized inflation rate, selects a universe and determines white holes for it. Through this mechanism, exploration is done. To perform exploitation, each universe is considered to have wormholes which connects them through a tunnel to the best universe formed so far in order to exchange objects (parameters). For this, each parameter is represented as below:

$$x_i^j = \begin{cases} \begin{cases} X_j + TDR \times ((ub_j - lb_j) \times r4 + lb_j), & r3 < 0.5 \\ X_j - TDR \times ((ub_j - lb_j) \times r4 + lb_j), & r3 \geq 0.5 \end{cases}, & r2 < WEP \\ x_i^j, & r2 > WEP \end{cases} \quad (35)$$

where X_j represents the jth parameter of the best universe formed so far, TDR (Travelling Distance Rate) and WEP (Wormhole Existence Probability) are coefficients, lb_j and ub_j denotes the lower bound and upper bound of the jth parameter respectively, x_i^j represents the jth parameter of the ith universe and r2, r3, r4 are random values in [0,1]. The formulas for the coefficients are as follows:

$$WEP = min + l \times (\frac{max - min}{L}) \quad (36)$$

where min and max are the minimum and maximum respectively, l tells the current iteration and L is the maximum iterations.

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \quad (37)$$

where p denotes the exploitation accuracy over the iterations. Higher values of p gives accurate exploitation/local search. Multi-verse optimizer is described in Table A.9.

MVO has been implemented to solve project scheduling problem [159], enhance kernel extreme learning machine for medical diagnosis [160], model solar radiation [161], solve economic dispatch problem [162]. In the area of robotics, researchers have implemented it to solve path planning in 3D search space [163], tune PID controller [164,165], plan navigation of quadrotors [166], plan path of mobile robot [167]. Only a few researchers have applied MVO for navigation of smart vehicles as outlined in Table 10

.

Table 10: Multi-Verse Optimizer for path planning of smart vehicles

Type	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
Evolutionary Multi-Verse Optimizer	Single			Python (3.7)	• Parameters of each solution are the weights and bias of implemented Multi-Layer perceptron Network.	[167]
MMVO	Single	Static	Geometrical (2D and 3D)		• 3D path planning in a modelled 3D environment is examined.	[163]

MMVO: Modified Multi-Verse Optimizer;

2.2.10 Bat Algorithm (BA)

Bat algorithm, an incite from microbat's sonar behaviour in communication was developed by Xin-She Yang [168]. Microbats release a short but loud bust of sound at frequencies within 25 kHz and 150 kHz and pays attention to the reverberated sound from near objects. The nature of the echoed sound helps the bat in determining the location of objects, thus defining their echolocation behaviour. Sometimes, the frequency of pulse emission and loudness of sound increases when they are in search of prey and decreases when a prey is found. The bat algorithm has been designed using idealized rules based on the echolocation behaviour of microbats.

- It is assumed that bats estimates distance from reverberated sound and are able to contrast prey/food from other entities.

- Bats move arbitrarily towards a prey at positions x_i with velocity v_i, frequency f_{min}, wavelength λ and loudness A_0.
- The loudness is assumed to be between a large positive value and a minimum defined value A_{min}.

For the bat algorithm, the frequency which is assumed to be within $[0, f_{max}]$, new velocity and new position are defined below.

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{38}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i \tag{39}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{40}$$

where x_* represents the current global best solution from all n bats and β is a vector within [0,1]. The loudness (A) is initially assumed to be any positive number typically in the range [1,2] and then updated by a constant $\alpha \in [0,1]$ as seen in equation 41. $A = 0$ when a solution is found. The rate of pulse emission $\boldsymbol{r_i^0} \in [0,1]$ is controlled by a constant γ which can have the same value as α.

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)], \tag{41}$$

For local search, new solutions for each bat are calculated using random walk once a solution is picked from the current best ones.

$$x_{new} = x_{old} + \epsilon A^t \tag{42}$$

where $\epsilon \in [-1,1]$ denotes a random number in $[-1,1]$ and A^t represents average loudness of all bats at time t. Table A.10 narrates the pseudocode of bat algorithm.

Implementation of bath algorithm in the field of mobile robot's planning and navigation has shown tremendous results as seen in various literatures. Table 11 lists some studies that have proposed improvements on BA with the variables considered

in the study. Fatin et. al [169] proposed a modified frequency bat algorithm (MFB) to observe its performance in obtaining a shortest path from initial to end point when compared to the standard bat algorithm. When a robot detects moving obstacles, this novel algorithm combines obstacle detection and avoidance techniques and relies on sensor data to plan a new path. Simulation was done on a grid mat environment, and results showed it outperformed the standard bat algorithm.

Table 11: Bat algorithm for path planning of smart vehicles

Type	Population size	A(0)	r(0)	α	γ	f_{min}	f_{max}	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
MFB	5	1	0.5	0.98	0.8	0	10	Single	Dynamic	12 x 12 Grid	MATLAB	• Obstacle detection and avoidance method is integrated in the algorithm.	[169]
Type-1 FLS-BA	20							Single	Static			• BA modifies Type-1 FLS to generate optimum trajectory. • Proposed method aims at obtaining the least mean square error in trajectory tracking.	[170]

MFB: Modified Frequency Bat algorithm; FLS-BA: Fuzzy Logic System – Bat Algorithm

2.2.11 Tabu Search (TS)

Tabu search [171] is an optimization algorithm that uses constraints to avoid local minima in a given search space. It employs flexible memory cycles which intensify and diversify local search patterns in order to obtain a suitable solution. During exploration process, all information about current solution and explored solutions are tracked [172]. Tabu search uses neighbourhood search methods to navigate from a solution x to a feasible solution x' which is in the neighbourhood of x. This is done iteratively until a stopping criteria terminates the process. Tabu search explores the neighbourhood of each solution using memory structures that stores visited solutions. This technique aids the search process to avoid being trapped in local minima and explores other solutions in the search space [173]. These memory structures, also known as tabu list contains a set of banned solutions that have just been visited n iterations ago (n – known as tabu tenure is the number of iterations stored in the tabu list). Implementing Metaheuristic Algorithms describes the procedure for implementing this algorithm.

Only a few studies have been conducted on the use of the tabu search algorithm for mobile robot navigation. Some of these studies as seen in Table 12 have concluded on new hybrids that outperform the basic tabu search algorithm in path planning problem. [174] proposed a novel tabu search algorithm for routing multiple Automated Guided Vehicles in a warehouse. [175] designed a tabu search system model to solve the problem of global path planning on grid maps. [176] developed a modified tabu search method to navigate mobile robot in a complex environment. [177] fused the rules of tabu search into a designed fuzzy controller to solve online navigation problems. [178]

recommended a hybrid algorithm of tabu search and particle swarm optimization to obtain best path for autonomous mobile robots.

Table 12: Tabu Search Algorithm for path planning of smart vehicles

Type	Iteration number	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
TS-PATH	30	Single	Static	Grid (see Figure 11)	Designed a C++ simulation model	• The effectiveness of the tabu search for the global path planning problem is investigated.	[175]
PSO-TABU	31	Multiple	Static	Geometrical	Designed a C simulation environment	• In terms of solution quality and computation times, PSO-TABU, outperforms the basic PSO and TABU search algorithm.	[178]
Modified Tabu Search		Single (Khepera-III robot)	Static	10 x 8cm Grid (simulation) and 400 x 300cm^2 Geometrical (experimental)	V-REP simulation software	• Algorithm is verified in both simulation and experimental platform. • Deviation between the simulation and experimental results is about 4%.	[176]
ANFIS		Single	Static	10 x 9, 10 x 10 and 10 x 14 Geometrical	MATLAB R2010b	• Heuristic rules of Tabu search is infused in fuzzy controller. • Fuzzy planner can handle online navigation task.	[177]
Tabu Search	9	Single	Static	10 x 10 Grid	MATLAB	• Algorithm generates trajectories to multiple goals	[179]

Type	Iteration number	Type of vehicle	Type of obstacles	Type of Map	Software	Remark	Ref
						using the shortest possible path.	
GSTIACA		Multiple	Dynamic	Grid	e-Puck architecture in the Webots simulation environment.	• Real-time simulation shows concurrent navigation and map building in dynamic environments.	[180]
TS/FA	5 - 7	Single	Static	561 x 380 px^2 and 433 x 430 px^2 Grid	MATLAB 2014a and V-rep simulator	• TS/FA is an offline hybrid algorithm. • Bezier curve is used to smoothen generated path.	[181]

TS-PATH: Tabu Search Path; PSO-TABU: Particle Swarm Optimization – Tabu Search; ANFIS: Adaptive Neuro-Fuzzy Inference System; GSTIACA: Genetic Shared Tabu Inverted Ant Cellular Automata; TS/FA: Tabu Search / Firefly Algorithm

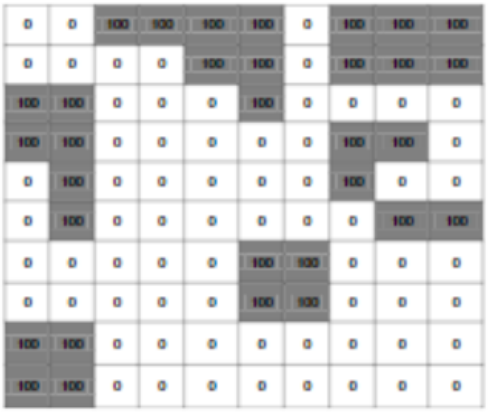


Figure 11: Sample maps implemented for tabu search algorithm: TS-PATH grid map [175]

2.3 Analysis

It is worth noting that all the metaheuristic algorithms used in this work have been validated on a variety of benchmark functions. Benchmark testing sets are mathematical functions that help to realize solutions of dimension d, which give global optima values [182]. Most common benchmark functions as seen in Table 13 are either unimodal, multimodal or combinatory (combines unimodal and multimodal). Unimodal functions give a single optima solution while multiple optimum solutions are obtained using multimodal functions.

Metaheuristic algorithms play an important part in the research field in proffering solutions to real world problems owing to benefits stated by [183]. Its high effectiveness and flexibility makes it useful in solving increasing complex problems. Citations are used to gauge a metaheuristic algorithm's popularity. Table 14 shows the ranking of the aforementioned algorithms based on number of citations.

Table 13: Common benchmark problems [117,150,182]

Function Name	Equation	Objective value	Modality
Spherical	$\sum_{i=1}^{i=d} {x_i}^2$	0	Unimodal
Schwefel 2.22	$\sum_{i=1}^{i=d} \lvert x_i \rvert + \prod_{i=1}^{i=n} \lvert x_i \rvert$	0	Multimodal
Schwefel 2.21	$\max_{1 \le i \le n} \lvert x_i \rvert$	0	Unimodal
Rosenbrock	$\sum_{i=1}^{i=d} 100({x_{i+1}} - {x_i}^2)^2 + (1 - x_i)^2$	0	Multimodal
Step	$\sum_{i=1}^{d} \lvert x_i + 0.5 \rvert^2$	0	Unimodal
Schwefel	$418.9829\mathrm{d} - \sum_{i=1}^{i=d} -x_i \sin\sqrt{\lvert x_i \rvert}$	0	Multimodal
Rastrigin	$10 * d + \sum_{i=1}^{i=d} {x_i}^2 - 10\cos(2\pi x_i)$	0	Multimodal
Auckley	$-20 * \exp\sqrt{\frac{1}{d}\sum_{i=1}^{i=d} {x_i}^2} - \exp\left(\frac{1}{d}\sum_{i=1}^{i=d} \cos(2\pi x_i)\right) + e$	0	Multimodal
Griewank	$\sum_{i=1}^{d} \frac{{x_i}^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0	Multimodal

Table 14: Citation ranking of algorithms used in this work (Retrieved 28 Nov, 2022, Google Scholar)

Rank	Year	Algorithm	Number of citations
1	1995	Particle Swarm Optimization [61]	75041
2	1975	Genetic Algorithm (GA) [29]	74165
3	1992	Ant Colony Optimization [46,184]	5685 (from 1992)

			15447 (from 2006)
4	2014	Grey Wolf Optimization [142]	9881
5	1986	Tabu Search Algorithm [171,185]	6320 (from 1986) 9716 (from 1989)
6	2005	Artificial Bee Colony [186]	8060
7	2009	Cuckoo Search Algorithm [103]	7217
8	2016	Whale Optimization Algorithm [122]	6649
9	2008	Firefly Algorithm [89]	6224
10	2016	Multi-Verse Optimizer [153]	1656

Choosing an algorithm to implement for path planning problems is important when considering the application of smart vehicles as listed in Table 15. The type of algorithms matters when planning smart vehicles for rescue missions and urgent task compared to when planning for surveillance or logistic purposes. Exploration and exploitation are always trade-offs in optimization problems. The exploration process efficiently searches the search space and avoids local optima regions to obtain global solutions. But the downside to this is a low convergence speed. Exploitation process results to high convergence speed, but there's a possibility of getting trapped in local optima regions. Therefore, in choosing an algorithm to perform path planning for a particular task, it is important to consider these parameters - convergence speed and finding global optimum.

2.3.1 Analysis On Computational Time and Shortest Path

A hybrid cuckoo search obtains an optimum path in less time when compared to PSO and GA [187]. PSO produce a better convergence result than BA in tuning omnidirectional mobile robot [188]. In addition, a hybrid PSO provides a shorter path in reduced time when compared to a modified BA and ABC [189].

CS is proven to outperform BA in finding an optimum path [121]. A multiobjective GWO achieves a better overall coverage of the search space to obtain an optimum path

than multiobjective PSO and GA [190]. As part of a comparison to a proposed method, ACO performs better in obtaining optimum path than GA [191]. Robot path produced by a hybrid ACO-PSO is more optimal than ACO [192]. Simulation results from ABC shows it obtains a shorter path than PSO [88]. Smart vehicles are used in industries accomplish so many activities especially logistics in warehouses, material handling in manufacturing factories, cleaning and disinfection, surveillance, rescue activities, military activities. The studied algorithms are best fitted for some of these task shown in Table 15.

Table 15: Various tasks performed by smart vehicles

Tasks
Vehicle scheduling
Warehouse material handling
Unmanned ground vehicle (military)
Search operation
Security and surveillance
Cleaning and disinfection operation
E-commerce delivery

2.4 Simulation Platform

Simulation of metaheuristic algorithms are performed on various platforms that offer mathematical and graphical functionalities. While some offer a graphical view of a simulated output, others like gazebo-ROS provide a 3D animated environment to visualize a simulated output. The most common platform used by researchers which is MATLAB provide inbuilt functions to ease the programming of metaheuristic algorithms. Rather than using available simulation platforms, some researchers have built theirs using some basic programming languages like C, C++ [175,178]. Table 16 lists the available languages and simulation platforms used by researchers. [193] has done a quantitative comparison on some of these simulators.

Table 16: Common platforms and programming languages simulation.

Platform/Programming language	Remarks	Ref
Python	• High-level user-friendly programming language.	[194,195]
C, C++	• High-level programming language for general-purpose programming.	[196]
MATLAB	• Modelling and simulation software built by MathWorks.	[197]
CoppeliaSim (formally V-REP)	• Creates room for importing personally designed robots. • Robotic models can be controlled using C, python, or MATLAB scripts including ROS node.	[198–200]
ROS with MoveIt	• MoveIt is the primary simulator in ROS for motion planning, 3D perception, manipulation and control.	[201–203]
GazeboSim	• Offers various libraries and cloud services for robot simulation.	[204–207]
Webots	• Offers a complete development environment to simulate robots and mechanical systems.	[208–210]
MORSE	• Modular Open Robot Simulator Engine based on Blender game engine. • A 3D simulator that offers a set of standard sensors, actuators and robotic bases.	[211,212]
USARsim	• Urban Search and Rescue simulator for multi-robot purposes.	[213–215]

2.5 Summary

This study gives a comprehensive review on various metaheuristic algorithms and their hybrids developed by researchers to solve the problem of planning and navigating smart vehicles. Classification of these algorithm has been based on population-based (evolutionary, swarm intelligence, nature-inspired) and trajectory-based algorithms. A comprehensive description of each algorithm has been done followed by reviews on recent articles in the timeframe of 12 years (i.e. 2010 - 2023) of which majority of the reviewed articles are within 2017 to 2023. The major parameters considered during the review are the type of vehicles (single or multiple robot), type of obstacles (static

or dynamic), type of map (topological, geometrical or grid map) and simulation platform for analysis. Furthermore, analysis of these algorithms has been made based on computational time and finding shortest optimum path and tasks carried out by smart vehicles are enlisted.

It is observed that navigation of smart vehicles is a never ending problem because of the need to optimize path length from a start to a goal point in less time. It can be noted that researchers have done improvements and upgrades on some these algorithms to solve observed anomalies [118,150]. A key observation is that a lot of hybrids between these algorithms have been developed to fine tune some of its parameters [136] or merge their advantages to become more robust [119]. To better fit an obtained path to the kinematics of smart vehicles, path smoothening was considered as a parameter in some reviewed literatures. Cubic polynomial [85,91], Bezier curve [78,181], B-spline curve [56], Cubic Ferguson spline [88] among others generates smooth paths either by interpolating through the points along the path or by using control points to produce a smooth path close to the obtained linear path.

Majority of the reviewed work performed their simulation in a static environment, however considering the use cases of smart vehicles in dynamic environments (moving obstacles and humans), more work needs to be done in the navigation of smart vehicle in changing environments. One case study has shown that using object detection sensors and refining observed data through neural network with fine-tuned weights to manipulate a smart vehicle can be a solution to planning a path in changing environments [167]. Also, a hybrid of reinforcement learning with metaheuristic algorithms can be studied to solve path planning problems. An example can be incorporating reinforcement learning agents to monitor the trade-off of exploitation

and exploration of population-based metaheuristic algorithms when planning a navigation path.

Chapter 3

RESEARCH METHODOLOGY

For this research methodology as shown in Figure 12, simulations are performed on different maps to obtain the length of path from start to goal point and the time it takes to obtain the desired path. Different metaheuristic algorithms are considered and for each algorithm, the different sampling methods are applied. These samples are on different environment with varying obstacle size and position. The start and goal point is specified for each environment, simulation is performed and the time and path length are actualized.

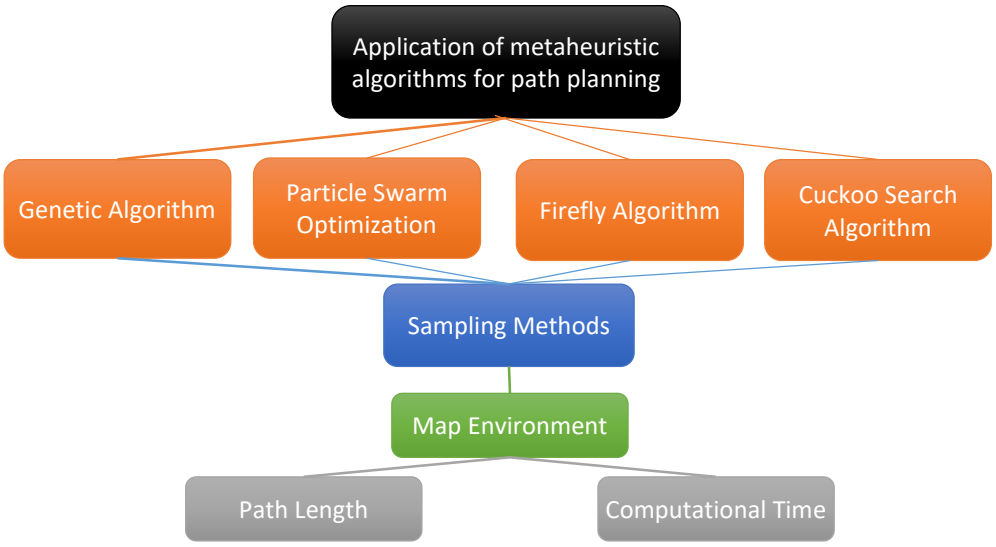


Figure 12: Research methodology

3.1 Different Maps Implemented

Looking through various literatures, it was observed that the source of their maps were not stated. After thorough research, a set of path planning benchmark maps which

contained both simple and complex obstacles were listed by Marco et. al [82]. About four of these maps are selected which can be seen in Figure 13.

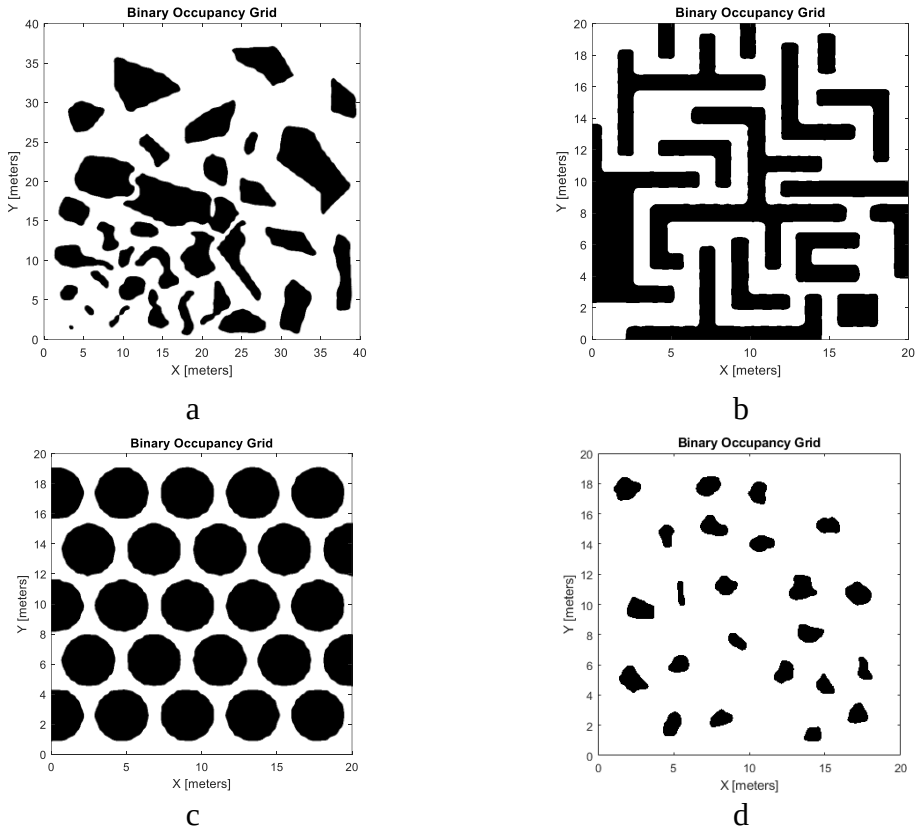


Figure 13: Binary occupancy grid maps: (a) 40 x 40 large map (b) 20 x 20 maze map (c) 20 x 20 rockpile (d) 20 x 20 pothole map.

3.2 Sampling Methods

To understand the effects of sampling on population size of metaheuristic algorithms, some sampling methods are considered.

3.2.1 Pseudo-Uniform Sampling

Probabilistic roadmap method is a sampling-based algorithm that solves robotic motion planning problems [216]. The basic probabilistic roadmap method (PRM) has two phase: learning and query phase [217]. Modified-PRM proposed by [218] utilized pseudo-random sampling method instead of random sampling to sample points on an environment.

This sampling method samples nodes along a reference axis which is the main spatial axis between the starting and goal node. With the order of the start and goal nodes represented as $S(x_s, y_s)$ and $G(x_g, y_g)$ respectively, the length L and declination of the spatial principal axis θ are calculated below.

$$L = \|G - S\| \tag{43}$$

$$\theta = \frac{\pi}{2} - \arctan\left(\frac{|y_g - y_s|}{|x_g - x_s|}\right) \tag{44}$$

Given n as the number of sample points, the longitudinal sampling spacing N_d is:

$$N_d = \frac{L}{n} \tag{45}$$

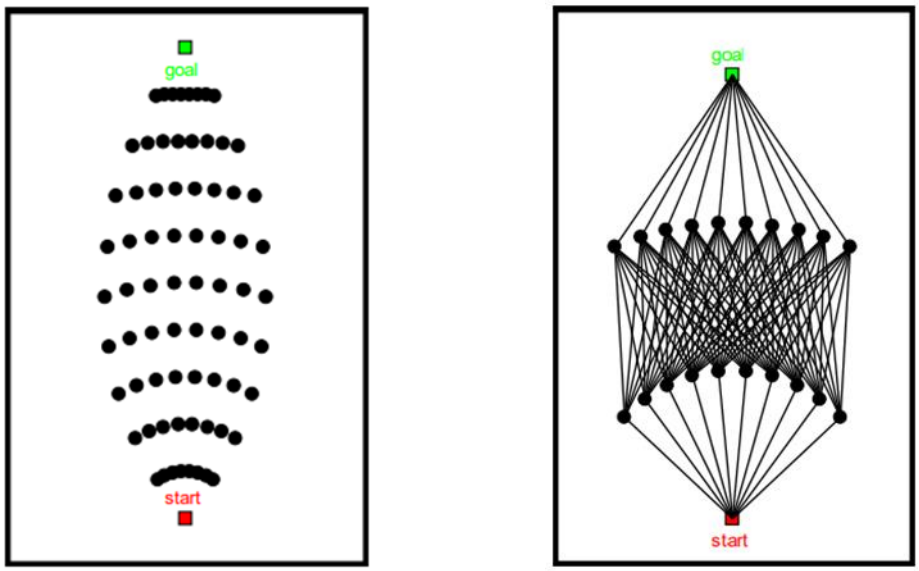
A sampling point $P_{i,j}(x, y)$ is calculated as follows:

$$x = x_s + r_d * \cos(\theta + \phi_j) \tag{46}$$

$$y = y_s + r_d * \sin(\theta + \phi_j) \tag{47}$$

$$r_d = i * N_d, \quad i = [1,2, \dots, n] \tag{48}$$

where (x_s, y_s) is the beginning point; r_d denotes the sampling radius which has its centre on the starting point; while $\phi_j \in [-\phi_m, \phi_m]$ is the sampling point's angle of deflection, ϕ_m represents the maximum deflection angle which determines the angle of the sampling region as shown in Figure 14a. For this work, only the uniform pattern of this sampling method is considered.



a b

Figure 14: Pseudo-uniform sampling (a) Sampling with $\phi_m = 20$. (b) Adjacent sampling layer connection.

Connecting the nodes, neighbouring layer connection strategy is used to connect adjacent sampling layers as shown in Figure 14b.

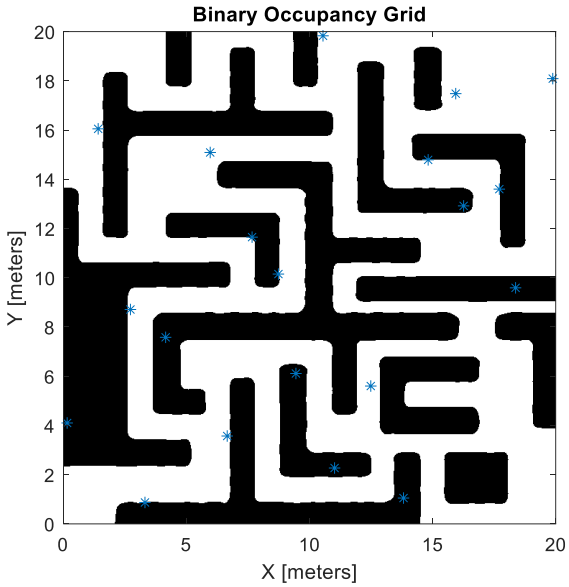
3.2.2 Random Sampling

Random sampling is the uniform distribution of random numbers within a given interval. For random sampling of points on a map, the points are generated within the boundaries of the map using the "rand" function of MATLAB. A value is generated and paired with another value to form a random point on the map. Choice of number of points to generate are based on simulations of different iteration of number of points on a map depending on its complexity (the area obstacles occupy with respect to the area of the map).

3.2.3 Latin Hypercube Sampling

This sampling distributes random points evenly over a sample space. It helps to generate controlled random samples and often applied in Monte Carlo analysis because it can reduce number of needed solutions to achieve accurate results. In an n-dimensional space, the search space is distributed evenly and samples are selected in

each space. In the context of this study, this sampling method generates samples on the x and y coordinate to form points on a map. A representation of this sampling is shown in Figure 15.



X	16.26	0.16	8.74	12.49	10.55	6.66	4.16	11.02	18.37	13.82	19.88	1.42	7.67	2.73	9.45	14.82	3.32	15.94	5.97	17.71
Y	12.92	4.1	10.15	5.6	19.83	3.57	7.58	2.27	9.59	1.05	18.09	16.05	11.65	8.71	6.11	14.79	0.87	17.48	15.09	13.6

Figure 15: Representation of Latin Hypercube sampling on a map.

3.3 Implementing Metaheuristic Algorithms

A comparative analysis of various metaheuristic algorithms is executed to obtain the shortest path and the computational time. A number of population-based algorithms are selected to analyse their performance based on the already defined parameters (distance and time). The population samples in these algorithms are taken to be different connected paths from a start node to a goal node while the fitness value is the path length. This implementation is done on the selected maps for this study. The paths are defined by the sampled points on the map. A description of how the selected algorithms (GA, PSO, FA, CSA) are implemented, variables considered and modifications made are revealed in this section.

3.3.1 Genetic Algorithm

The possible solution (chromosome) is the set of nodes belonging to a path and having the start and goal node. A chromosome is denoted by a number of gene. Every gene contains two values which are (x, y) coordinates of the graph as illustrated in Figure 16.

	Start						goal
x-coordinate	9	8	7	5	--- --- --- ---	2	3
y-coordinate	10	10	11	8		5	5

Figure 16: A chromosome

The sum of Euclidean distance between two genes in a chromosome defines the fitness function. Selection is done based on the best three fitness values using Roulette Wheel Selection i.e. chromosomes with the smallest path length. Crossover operator (crossover scattered) is implemented to generate a new offspring for the next generation. Chromosomes will be generated in the new generation together with the new offspring chromosome. After a number of generations, the program is terminated and the chromosome having the best value is chosen to be the optimum path.

3.3.2 Particle Swarm Optimization

Particles in PSO represent solutions with position and velocity. Considering that sample points on a map are discrete values, the basic PSO which deals with continuous values doesn't fit in to solve path planning problem, therefore a discrete PSO proposed by Quan-ke et. al [219,220] is implemented. It utilizes the inertia weight, cognitive and social factors for position update in each particle. Each particle follows its own position X_i^t to achieve it's personal best P_i^t and global G^t position. The below defines the position update.

$$X_i^t = c_2 \otimes F_3(c_1 \otimes F_2(w \otimes F_1(X_i^{t-1}), P_i^{t-1}), G^{t-1}) \quad (49)$$

where w is inertia weight, and c_1, c_2 are cognitive and social factors. Three components are embedded in the equation above, one being $\lambda_i^t = w \otimes F_1(X_i^{t-1})$ having F_1 as the mutation operator with probability of w. λ_i^t is the particle's velocity. If a random number $r \in [0,1]$ is less than w, mutation is applied to produce λ_i^t, else $\lambda_i^t = X_i^{t-1}$. The second component $\delta_i^t = c_1 \oplus F_2(\lambda_i^t, P_i^{t-1})$ has F_2 as the crossover operator with probability c_1. If r is less than c_1, the crossover operator is applied to produce $\delta_i^t = F_2(\lambda_i^t, P_i^{t-1})$, else $\delta_i^t = \lambda_i^t$. The third component $X_i^t = c_2 \oplus F_3(\delta_i^t, G^{t-1})$ has F_3 as the crossover operator with probability c_2. If r is less than c_2, the crossover operator is applied to produce $X_i^t = F_3(\delta_i^t, G^{t-1})$, else $X_i^t = \delta_i^t$. The position is compared to its personal and overall best at each iteration. The optimum path is obtained when the difference between the current and previous global best value is below a given threshold or when a number of desired iterations are achieved.

3.3.3 Firefly Algorithm

Each firefly represents a possible path to be optimized. While a firefly's attractiveness in continuous problems is calculated with equation 9, the firefly's brightness or path, that is, its attractiveness in this study is defined by the fitness value of a path. Therefore, all other parameters considered in the equation is left out because of the discrete nature of the path. The distance between each firefly's position is represented by a mutation operation rather than the formula in equation 10.

3.3.4 Cuckoo Search Algorithm

The nests where the cuckoo lays its egg represents solutions to path planning problem. Each solution is a discrete path defined with start and goal points at both end, passing through other points that define the path. The operation of the discrete from of CSA for path planning problem stirs from the work of Aziz et. al [221] on travelling sales

person and the author of the algorithm, Xin et. al [89]. The survival of cuckoo's eggs in a nest defines the fitness value of such nest. Search for new nest is carried out by levy flight, which is calculated using Mantegna algorithm [222]. Choice of values for parameters such as the step-size scaling factor α, switching probability P, and levy exponent β are based on the values determined in the author's work [89]. Since the solutions are in discrete form, search for new solution is done by mutating each solution in an iteration with the best solution from previous iteration. This mutation operation is influenced by the step-size obtained in the levy flight calculations.

Chapter 4

RESULTS AND DISCUSSIONS

Simulations with different algorithms are performed on different maps with different sampling methods. First of all, a map is selected, a sampling method is chosen for that map, and all the metaheuristic algorithms are applied on the samples points generated. The sampling of points on the map greatly influences the form of path generated for each algorithm. The same start and goal point is used for all algorithms to understand and analyse their outcomes based on path length and computational time. Three runs are performed to observed a pattern of solutions obtained for each algorithm. All simulations are performed on MATLAB R2022b using a windows 11 pro (version 22H2) computer with these specifications: intel core i5-2450M CPU @ 2.50GHz CPU, 8GB RAM.

All metaheuristic algorithms used in this study require paths with same length size. Crossover and mutation especially in genetic algorithm can be easily performed when all the paths have same length size. To obtain paths with same length size with start and goal points at both ends of the path on random and Latin hypercube sampling is not feasible. Given obstacles with different geometry and their location on an environment, different paths to goal point from start point can be obtained with different length size. Therefore, A* algorithm is used to obtain an initial single solution. With this single solution, other solutions are generated to have same length size as the initial solution. The time taken to sample points and generate solutions on

a map for all sampling methods is considered differently from the time taken to optimize these solutions with metaheuristic algorithms in this study.

Same population size (50) and maximum iteration/generation number (100) were used for all algorithms. The population size was chosen from the suggestion of Yang et. Al's work [89] based on the author's studies and observations. Although the maximum iteration was set, iterations/generation stalled at values between 20 and 30.

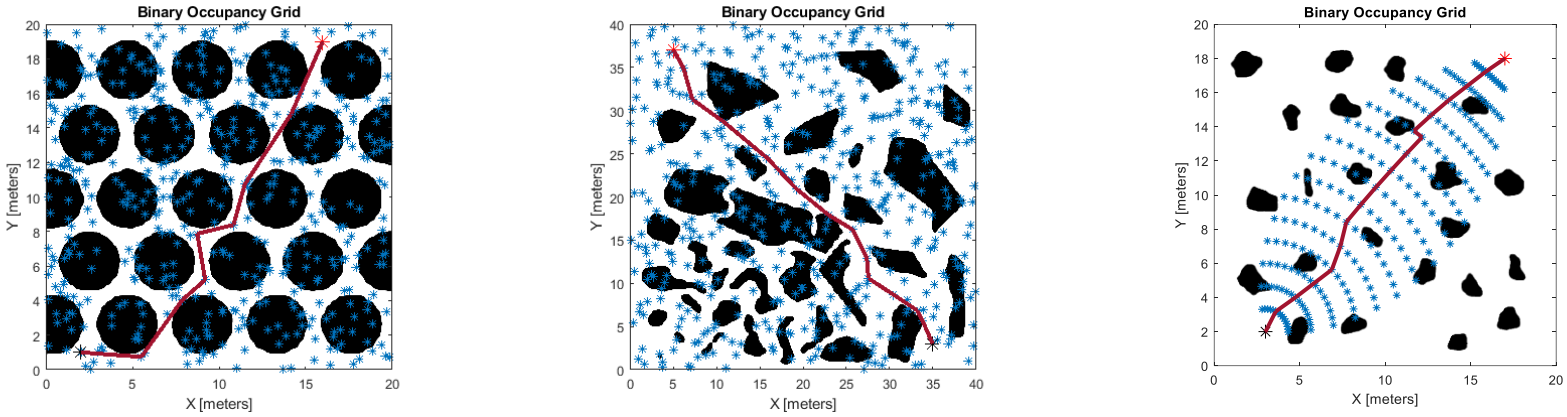
4.1 Results of GA

In the implementation of GA, and elite count (i.e. number of parents to move to the next generation) was 3. The crossover operator was “scattered”. The tabulated results of GA for all maps with different sampling methods are in Table 17. Different sampling methods on different map for GA are shown in Figure 17.

Table 17: Tabulated results of Genetic Algorithm

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
Random Sampling	40 x 40 large	(3,35)	(37,5)	49.0	5.4	37.6	52.4	5.5	37.9	53.3	6.2	39.2
	20 x 20 maze	(2,19)	(19,2)	47.1	13.4	124.5	46.5	18.0	124.1	47.0	16.5	108.7
	20 x 20 rockpile	(1,2)	(19,16)	26.1	5.6	100.7	24.7	5.3	94.8	25.5	4.5	94.8
	20 x 20 potholes	(2,3)	(18,17)	21.6	14.6	132.9	21.5	26.6	119.6	21.6	19.8	124.4
Latin Hypercube Sampling	40 x 40 large	(3,35)	(37,5)	47.2	13.5	34.9	51.5	11.8	37.0	47.3	8.7	40.3
	20 x 20 maze	(2,19)	(19,2)	46.6	14.4	113.4	44.0	26.4	97.8	45.2	22.5	106.7

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
	20 x 20 rockpile	(1,2)	(19,16)	26.3	41.4	125.4	25.2	6.9	105.0	25.0	5.2	106.0
	20 x 20 potholes	(2,3)	(18,17)	21.6	12.5	137.1	21.4	13.3	121.7	21.3	11.4	121.7
Pseudo-uniform sampling	40 x 40 large	(3,35)	(37,5)	49.3	8.8	70.4	49.3	6.6	61.6	49.3	7.1	61.9
	20 x 20 rockpile	(1,2)	(19,16)	25.4	2.3	40.8	25.4	2.1	42.0	25.4	2.0	41.1
	20 x 20 potholes	(2,3)	(18,17)	22.4	7.5	126.9	22.3	7.1	112.5	21.6	5.1	111.2



a b c

Figure 17: Different sampling methods on different map for GA (a) Random sampling on rockpile map (b) Latin hypercube sampling on large map (c) Pseudo-uniform sampling on pothole map

4.2 Results of PSO

The key parameters of PSO which are inertia weight, cognitive factor and social factor were chosen to be 0.8, 0.5, and 0.8 respectively. The crossover operator was “scattered” while the mutation operator was “Random resetting”. The tabulated results for all maps with different sampling methods are in Table 18.

Table 18: Tabulated results of PSO

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
Random Sampling	40 x 40 large	(3,35)	(37,5)	49.0	4.3	37.6	52.4	5.0	37.9	53.3	4.9	39.2
	20 x 20 maze	(2,19)	(19,2)	47.1	17.5	124.5	46.5	14.8	124.1	47.0	15.5	108.7
	20 x 20 rockpile	(1,2)	(19,16)	26.1	4.1	100.7	24.7	3.8	94.8	25.5	3.6	94.8
	20 x 20 potholes	(2,3)	(18,17)	21.6	7.9	132.9	21.5	13.8	119.6	21.6	11.2	124.4
Latin Hypercube Sampling	40 x 40 large	(3,35)	(37,5)	47.2	6.3	34.9	51.5	7.0	37.0	47.3	4.9	40.3
	20 x 20 maze	(2,19)	(19,2)	46.6	16.8	113.4	44.0	24.9	97.8	45.2	28.7	106.7

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
	20 x 20 rockpile	(1,2)	(19,16)	26.3	5.4	125.4	25.2	4.3	105.0	25.0	4.0	106.0
	20 x 20 potholes	(2,3)	(18,17)	21.6	10.8	137.1	21.4	7.4	121.7	21.3	8.2	121.7
Pseudo-uniform sampling	40 x 40 large	(3,35)	(37,5)	49.9	7.8	70.4	49.9	5.4	61.6	49.9	6.0	61.9
	20 x 20 rockpile	(1,2)	(19,16)	25.4	3.1	40.8	25.4	3.5	42.0	25.4	2.7	41.1
	20 x 20 potholes	(2,3)	(18,17)	21.6	5.9	126.9	21.7	5.9	112.5	21.7	4.5	111.2

4.3 Results of FA

The mutation operator implemented in FA was “Random resetting”. The tabulated results for all maps with different sampling methods are in Table 19.

Table 19: Tabulated results of Firefly Algorithm

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
Random Sampling	40 x 40 large	(3,35)	(37,5)	49.0	1.9	37.6	52.4	3.2	37.9	53.3	2.6	39.2
	20 x 20 maze	(2,19)	(19,2)	47.1	35.5	124.5	46.5	53.8	124.1	47.0	38.8	108.7
	20 x 20 rockpile	(1,2)	(19,16)	26.1	0.3	100.7	24.7	1.0	94.8	25.5	1.8	94.8
	20 x 20 potholes	(2,3)	(18,17)	21.6	1.4	132.9	21.6	2.0	119.6	21.7	3.3	124.4
Latin Hypercube Sampling	40 x 40 large	(3,35)	(37,5)	47.2	1.8	34.9	51.7	3.5	37.0	47.3	2.1	40.3
	20 x 20 maze	(2,19)	(19,2)	46.6	33.5	113.4	44.0	54.2	97.8	45.2	32.5	106.7
	20 x 20 rockpile	(1,2)	(19,16)	26.3	1.6	125.4	25.2	1.0	105.0	25.0	0.7	106.0

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
	20 x 20 potholes	(2,3)	(18,17)	21.8	1.8	137.1	21.6	3.8	121.7	21.4	3.0	121.7
Pseudo-uniform sampling	40 x 40 large	(3,35)	(37,5)	49.9	12.6	70.4	49.5	16.2	61.6	50.5	10.8	61.9
	20 x 20 rockpile	(1,2)	(19,16)	25.4	8.6	40.8	25.4	8.8	42.0	25.4	9.7	41.1
	20 x 20 potholes	(2,3)	(18,17)	22.5	10.4	126.9	23.0	15.9	112.5	22.8	11.2	111.2

4.4 Results of CSA

The key parameters step-size scaling factor α, switching probability P, and levy exponent β were chosen to be 0.25, 0.01, and 1.5 respectively. The mutation operator implemented in FA was "Random resetting". The tabulated results for all maps with different sampling methods are shown in Table 20.

Table 20: Tabulated results of CSA

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
Random Sampling	40 x 40 large	(3,35)	(37,5)	49.0	0.1	37.6	52.4	0.3	37.9	53.3	0.3	39.2
	20 x 20 maze	(2,19)	(19,2)	47.1	12.5	124.5	46.5	11.4	124.1	47.0	12.2	108.7
	20 x 20 rockpile	(1,2)	(19,16)	26.1	0.2	100.7	24.7	0.1	94.8	25.5	0.2	94.8
	20 x 20 potholes	(2,3)	(18,17)	21.6	0.1	132.9	21.5	0.1	119.6	21.6	0.2	124.4
Latin Hypercube Sampling	40 x 40 large	(3,35)	(37,5)	47.2	1.8	34.9	51.5	2.3	37.0	47.3	0.2	40.3
	20 x 20 maze	(2,19)	(19,2)	46.6	10.5	113.4	44.0	23.0	97.8	45.2	20.2	106.7

Sampling method	Map	Start	Goal	1st Run			2nd Run			3rd Run		
				Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)	Path length (m)	Time (s)	Sampling Time (s)
	20 x 20 rockpile	(1,2)	(19,16)	26.3	4.5	125.4	25.2	0.1	105.0	25.0	0.1	106.0
	20 x 20 potholes	(2,3)	(18,17)	21.6	0.2	137.1	21.5	0.1	121.7	21.3	0.1	121.7
Pseudo-uniform sampling	40 x 40 large	(3,35)	(37,5)	49.3	3.9	70.4	49.5	5.4	61.6	49.9	2.8	61.9
	20 x 20 maze	(2,19)	(7,6)	16.8	1.9	51.3	16.8	2.4	51.4	16.8	2.5	55.0
	20 x 20 rockpile	(1,2)	(19,16)	25.4	1.7	40.8	25.4	2.9	42.0	25.4	2.1	41.1
	20 x 20 potholes	(2,3)	(18,17)	21.7	2.1	126.9	21.6	2.6	112.5	22.1	2.3	111.2

4.5 Discussion

In the simulation results as tabulated above (i.e. Tables 17, 18, 19, 20), there are three runs for each operation is performed with a population size of 50. The nature of paths generated is as a consequence of where the sample points are located on the map. Figure 18 displays the Latin hypercube points on 40 x 40 large map. It can be observed that the sample points affect the pattern of the path. For the random sampling and Latin hypercube sampling, the total number of sample points is 500.

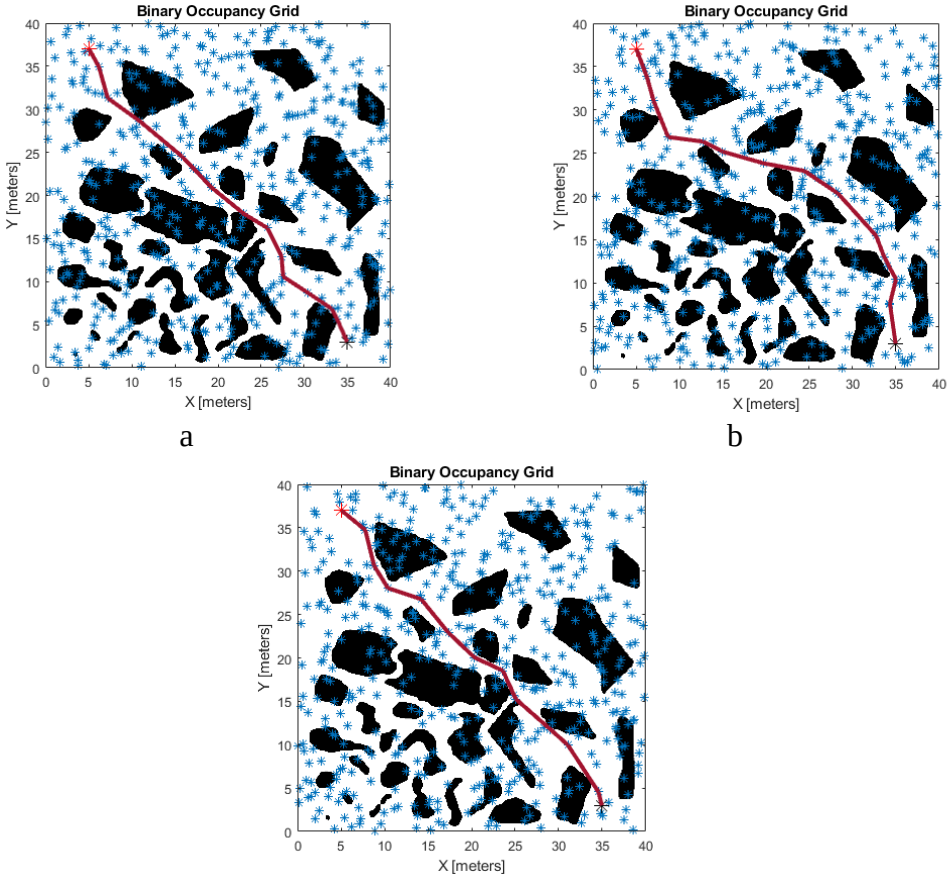


Figure 18: Latin Hypercube sampling on 40 x 40 Large map (a) First run (b) Second run (c) Third run

4.5.1 Comparative Analysis Between Algorithms

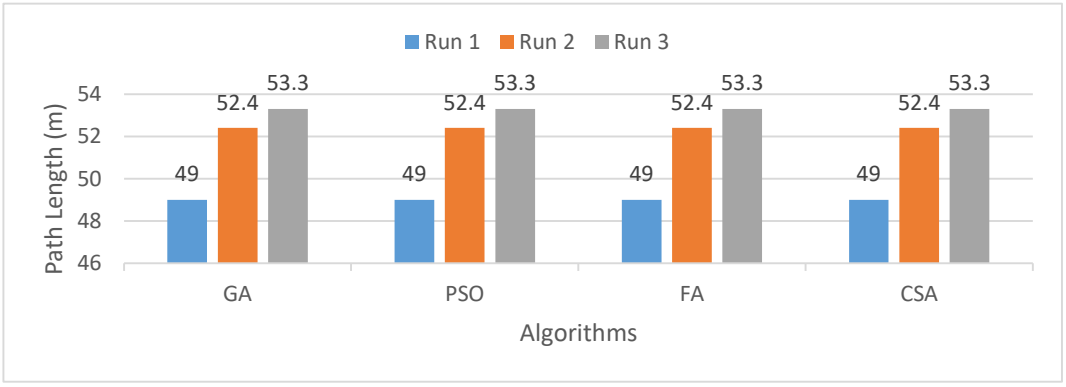
The selected algorithms are analysed based on the concept of "No-Free-Lunch (NFL)" theorem [223,224]. This theorem is used to measure the performance of different

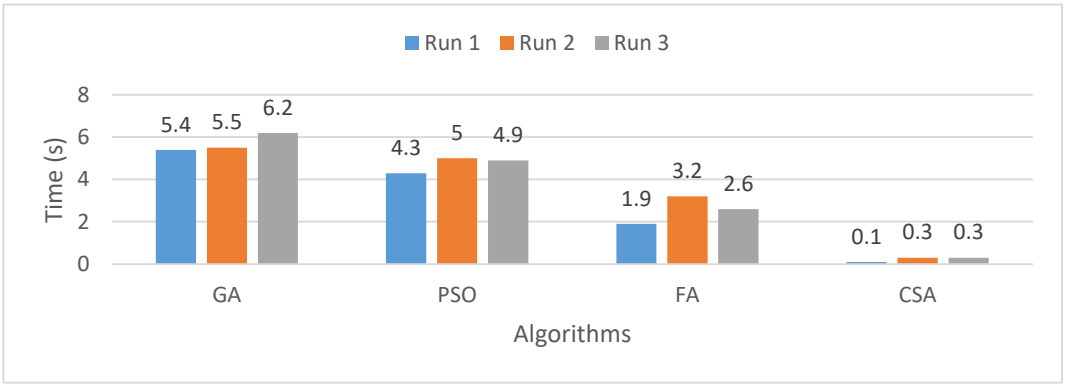
optimizations algorithms against each other on various problems. It explains the inability of an algorithm to outperform other algorithms on all represented objective functions, which means that there is no well-performing universal algorithm. If an algorithm performs well on average for one class of problem, then it must perform poorly on average over other problems. Drawing that into this study, the average results of each algorithm on path length is pitched against the average time for each algorithm. Thus, the performance of each algorithm is analysed against each other based on the path length and time.

In the case of Latin hypercube sampling on 40 x 40 large map, the results are tabulated in Table 21. The values on this table are excerpts from tables in sections 4.1 to 4.4 of this chapter. The graphical view of these results is displayed in bar charts in Figure 19. In applying the NFL theorem, the average of each algorithm across all sampling methods is obtained and the plotted bar chart is shown in Figure 20. The bar of GA and PSO shows GA giving a better result on path length, although PSO optimizes in less time thus meeting the conditions of NFL theorem. Although CSA outperforms all algorithms in terms of optimization time, it obtained same shortest distance as GA for run 1 and 2 and performed poorly against GA in the third run. But comparing the results through all three runs, CSA can be said to have performed better. The nature of paths generated by all four algorithms on large map with pseudo-uniform sampling is shown in Figure 21.

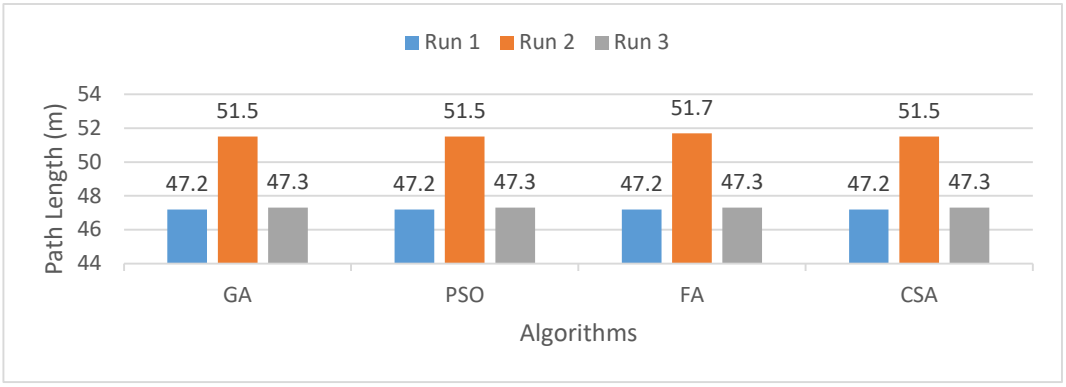
Table 21: Analysis of algorithms on 40 x 40 large map

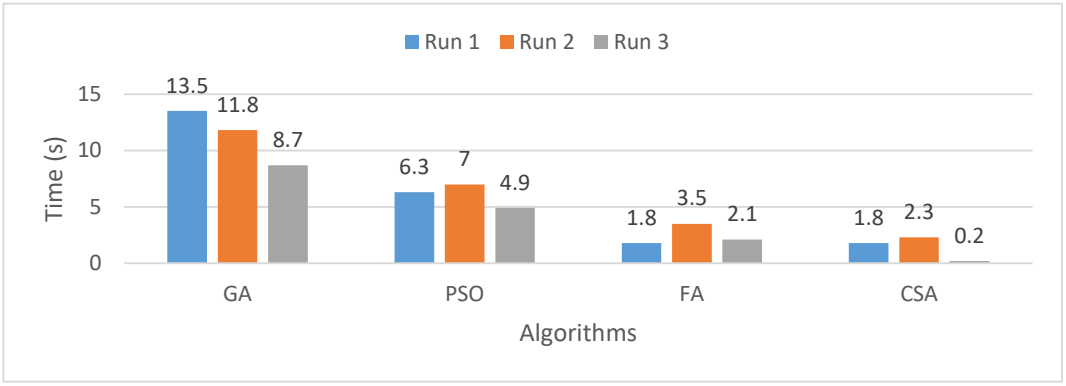
Algorithms	Run 1		Run 2		Run 3		Sampling
	Path length (m)	Time (s)	Path length (m)	Time (s)	Path length (m)	Time (s)	
GA	49.0	5.4	52.4	5.5	53.3	6.2	Random
PSO	49.0	4.3	52.4	5.0	53.3	4.9	
FA	49.0	1.9	52.4	3.2	53.3	2.6	
CSA	49.0	0.1	52.4	0.3	53.3	0.3	
GA	47.2	13.5	51.5	11.8	47.3	8.7	Latin Hypercube
PSO	47.2	6.3	51.5	7.0	47.3	4.9	
FA	47.2	1.8	51.7	3.5	47.3	2.1	
CSA	47.2	1.8	51.5	2.3	47.3	0.2	
GA	49.3	8.8	49.3	6.6	49.3	7.1	Pseudo-uniform
PSO	49.9	7.8	49.9	5.4	49.9	6.0	
FA	49.9	12.6	49.5	16.2	50.5	10.8	
CSA	49.3	3.9	49.5	5.4	49.9	2.8	



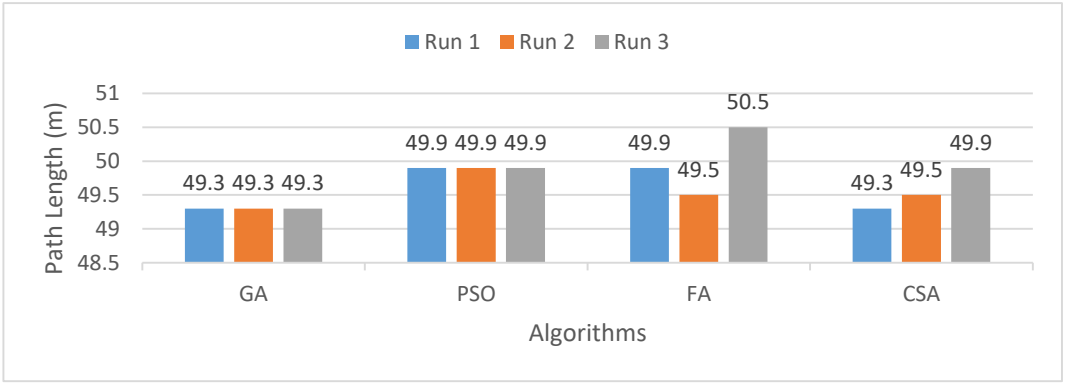


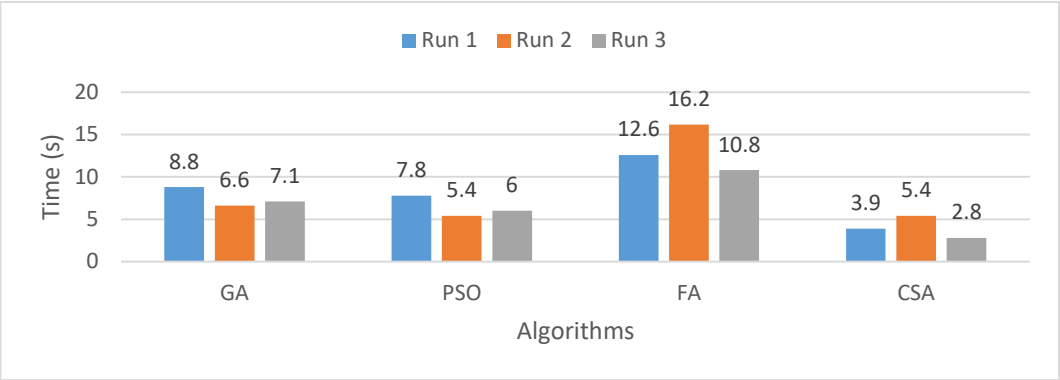
a


Run 1
Run 2
Run 3
Path Length (m)
54
52
50
48
46
44
47.2
51.5
47.3
47.2
51.5
47.3
47.2
51.7
47.3
47.2
51.5
47.3
GA
PSO
FA
CSA
Algorithms


Run 1
Run 2
Run 3
Time (s)
15
10
5
0
13.5
11.8
8.7
6.3
7
4.9
1.8
3.5
2.1
1.8
2.3
0.2
GA
PSO
FA
CSA
Algorithms

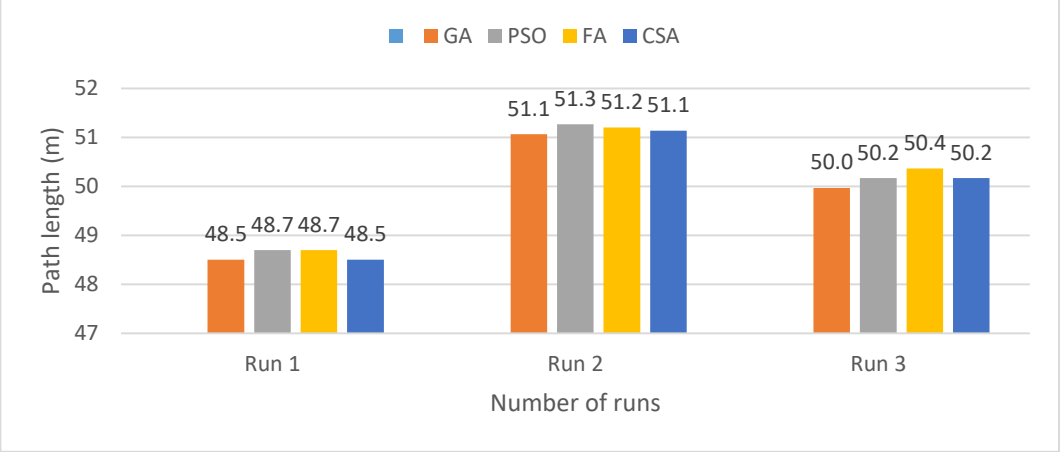
b


Run 1
Run 2
Run 3
Path Length (m)
51
50.5
50
49.5
49
48.5
49.3 49.3 49.3
49.9 49.9 49.9
49.9
49.5
50.5
49.3
49.5
49.9
GA
PSO
FA
CSA
Algorithms

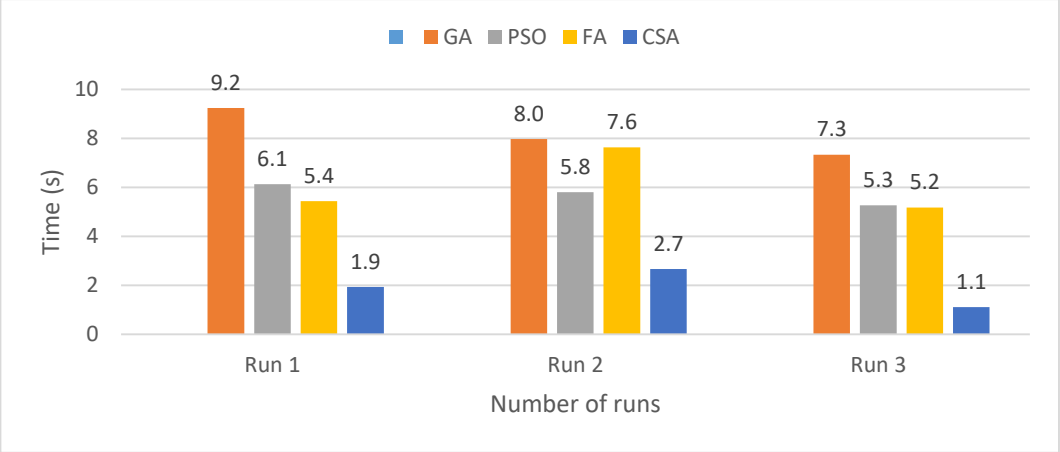


c

Figure 19: Bar chart analysis of algorithms considering path length and time on 40 x 40 large map (a) Random sampling (b) Latin hypercube sampling (c) Pseudo-uniform sampling



a



b

Figure 20: No-Free-Launch analysis of algorithms on 40 x 40 large map (a) Comparison on path length (b) Comparison on time

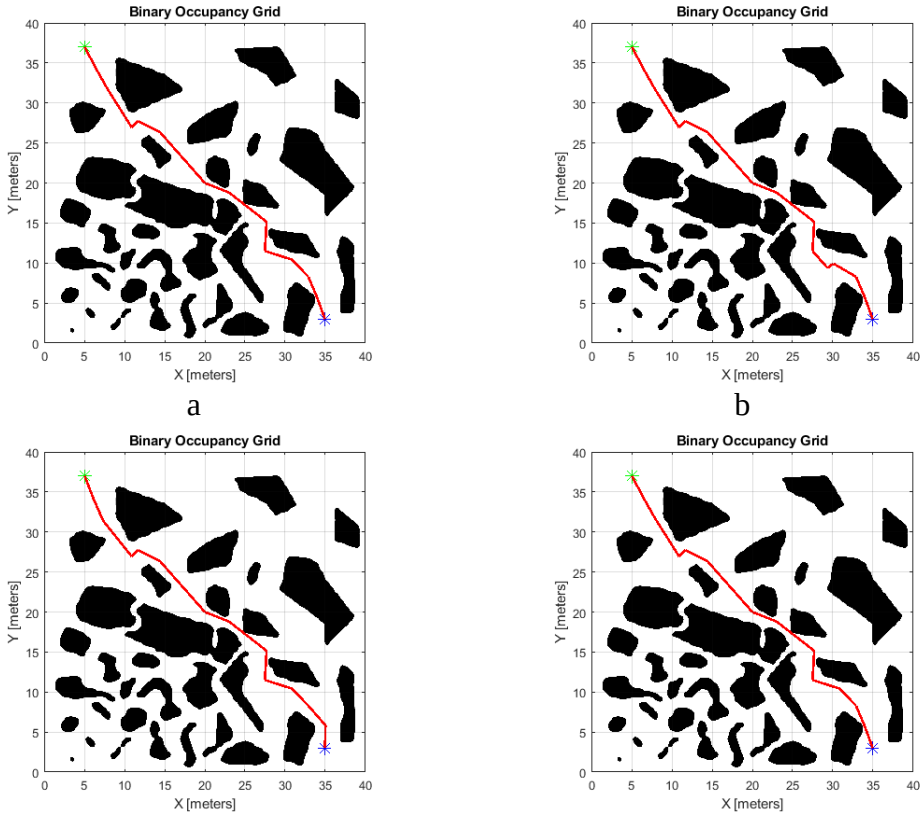
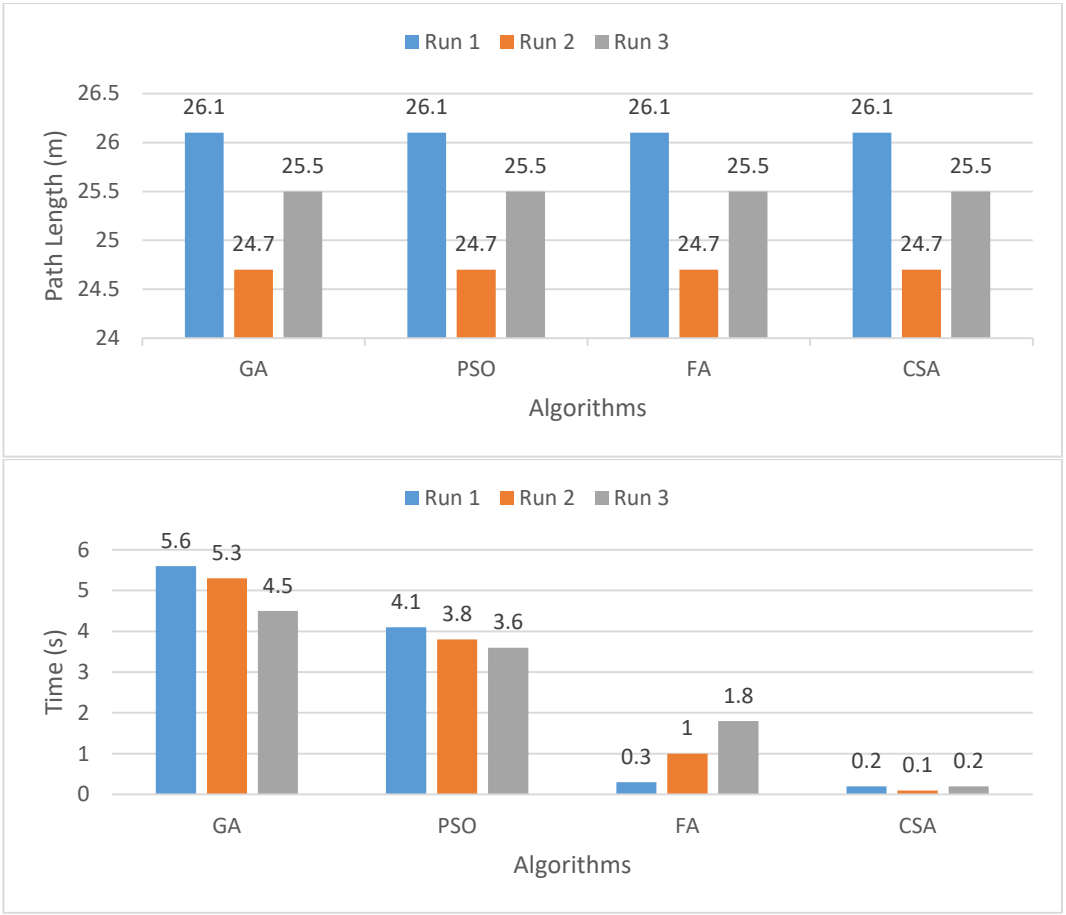


Figure 21: Pseudo-uniform sampling on 40 x 40 large map (a) GA (b) PSO (c) FA (d) CSA.

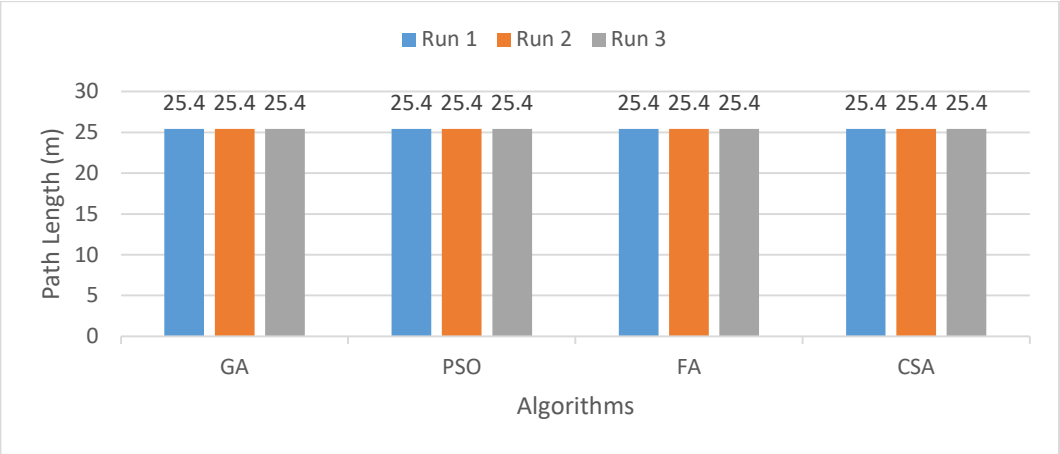
For a more complex map like the 20 x 20 rockpile map, due to the nature of points sampled within the configuration free space as observed in Figure 24, there's a high chance that there'll be more points on the obstacles than on the free space. This affects the results as all algorithm reveal same path distance for a single run in determined sampling method as observed in Table 22. The graphical view of these results is displayed in bar charts in Figure 22 Increasing the total sampled points on the map may give chance for the algorithms to have more points to optimize in generating a path, but that will be at the cost of computational time. Based on the result on path distance revealed in Figure 23, NFL theorem cannot be conceptualized as all algorithms obtained same shortest distance. It can be observed that CSA has the minimum optimization time compared to other algorithms.

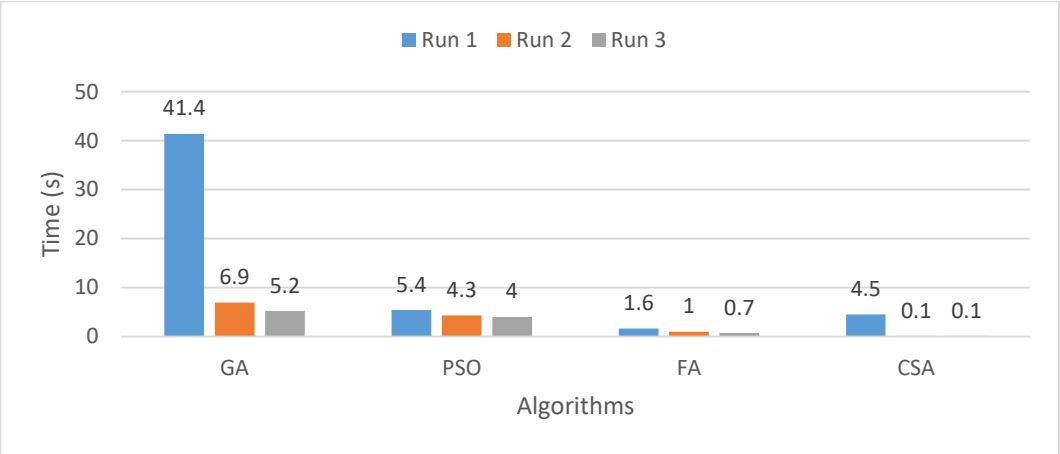
Table 22: Analysis of algorithms on 20 x 20 rockpile map

Algorithms	Run 1		Run 2		Run 3		Sampling
	Path length (m)	Time (s)	Path length (m)	Time (s)	Path length (m)	Time (s)	
GA	26.1	5.6	24.7	5.3	25.5	4.5	Random
PSO	26.1	4.1	24.7	3.8	25.5	3.6	
FA	26.1	0.3	24.7	1.0	25.5	1.8	
CSA	26.1	0.2	24.7	0.1	25.5	0.2	
GA	26.3	41.4	25.2	6.9	25.0	5.2	Latin Hypercube
PSO	26.3	5.4	25.2	4.3	25.0	4.0	
FA	26.3	1.6	25.2	1.0	25.0	0.7	
CSA	26.3	4.5	25.2	0.1	25.0	0.1	
GA	25.4	2.3	25.4	2.1	25.4	2.0	Pseudo-uniform
PSO	25.4	3.1	25.4	3.5	25.4	2.7	
FA	25.4	8.6	25.4	8.8	25.4	9.7	
CSA	25.4	1.7	25.4	2.9	25.4	2.1	

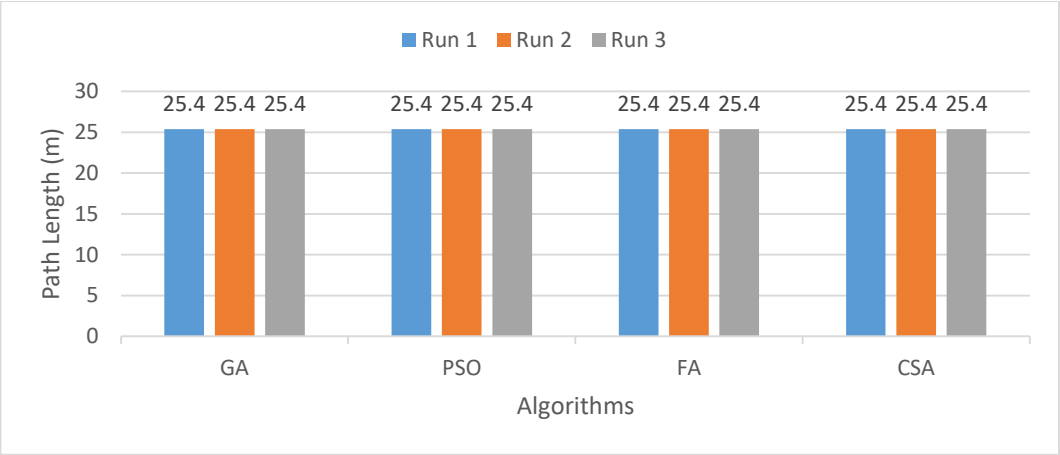


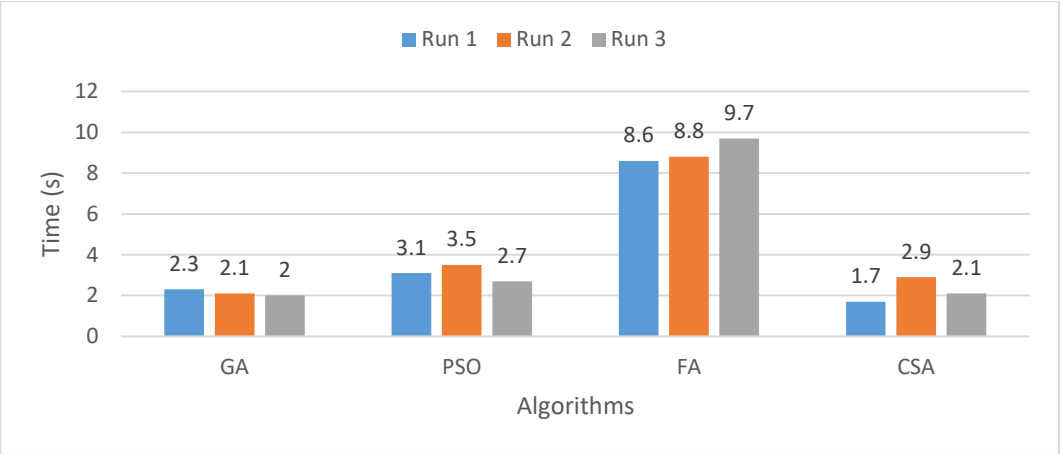
a


Run 1
Run 2
Run 3
Path Length (m)
30
25
20
15
10
5
0
25.4 25.4 25.4
25.4 25.4 25.4
25.4 25.4 25.4
25.4 25.4 25.4
GA
PSO
FA
CSA
Algorithms


Run 1
Run 2
Run 3
Time (s)
50
40
30
20
10
0
41.4
6.9
5.2
5.4
4.3
4
1.6
1
0.7
4.5
0.1
0.1
GA
PSO
FA
CSA
Algorithms

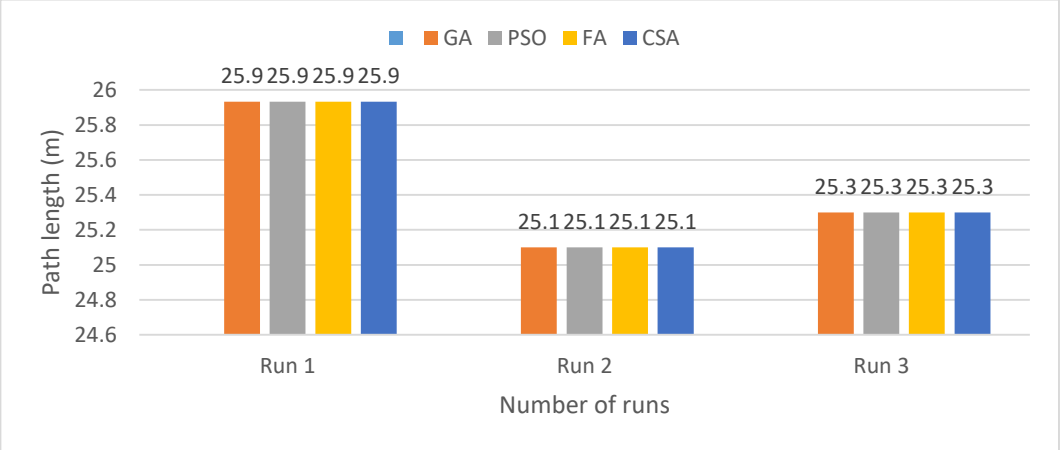
b


Run 1
Run 2
Run 3
Path Length (m)
30
25
20
15
10
5
0
25.4 25.4 25.4
25.4 25.4 25.4
25.4 25.4 25.4
25.4 25.4 25.4
GA
PSO
FA
CSA
Algorithms

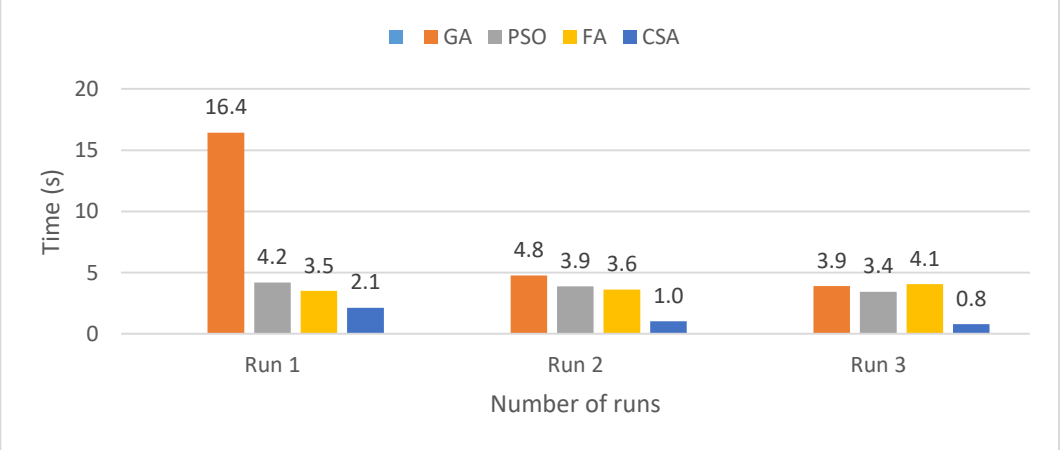


c

Figure 22: Bar chart analysis of algorithms on 20 x 20 rockpile map (a) Random sampling (b) Latin hypercube sampling (c) Pseudo-uniform sampling



a



b

Figure 23: No-Free-Launch analysis of algorithms on 20 x 20 rockpile map (a) Comparison on path length (b) Comparison on time

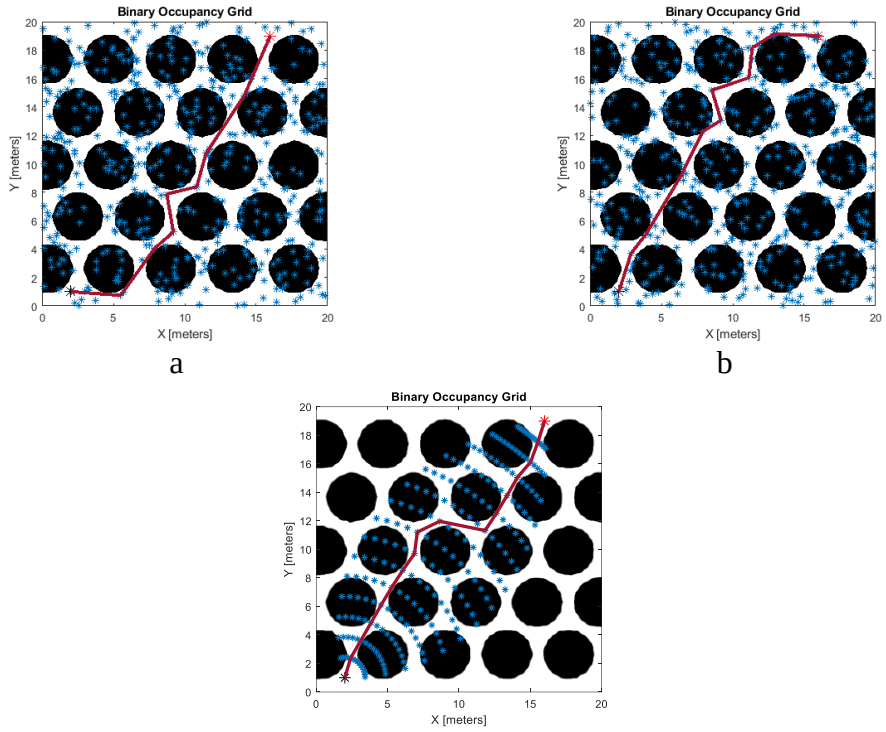
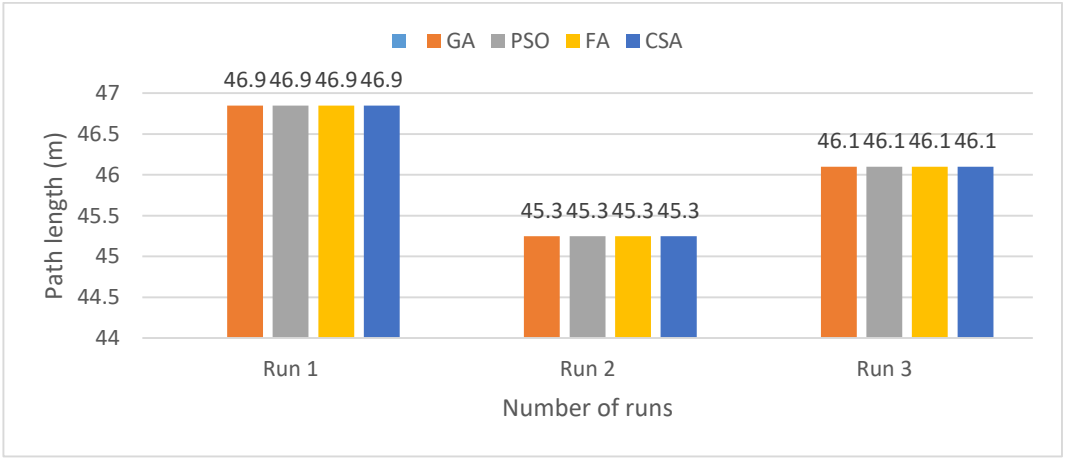


Figure 24: Disparity of sampled points (a) GA path on random sampling, run 1 (b) GA path on latin hypercube sampling, run 1 (c) GA path on pseudo-uniform sampling, run 1

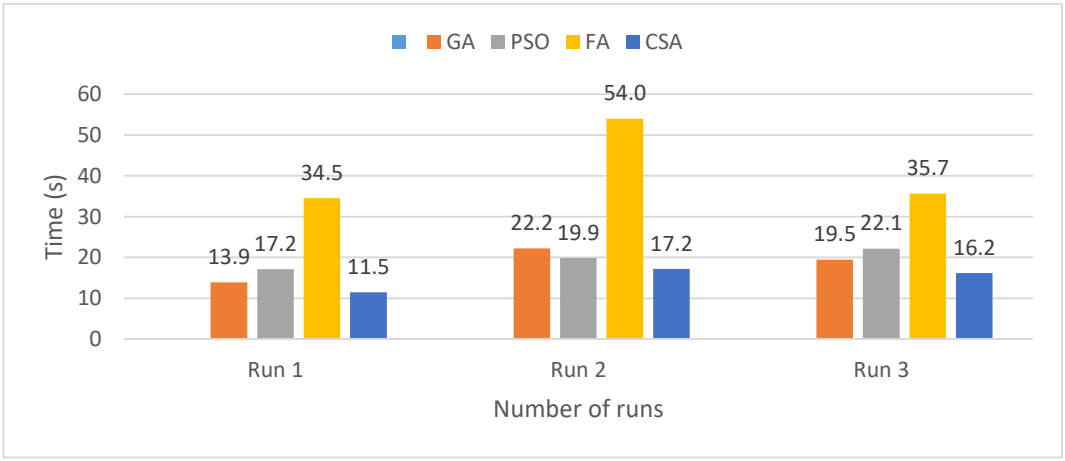
A similar problem in rockpile map can be observed in the maze map results in Table 23 as there are more narrow free configurations, therefore increase in total samples points means more sample points existing in free space. Based on the result on path distance revealed in Figure 25, NFL theorem cannot be conceptualized as all algorithms obtained same shortest distance for the three respective runs. It can be observed in Figure 25b that CSA has the minimum optimization time compared to other algorithms. The nature of paths generated with sampling methods is shown in Figure 26.

Table 23: Analysis of algorithms on 20 x 20 maze map

Algorithms	Run 1		Run 2		Run 3		Sampling
	Path length (m)	Time (s)	Path length (m)	Time (s)	Path length (m)	Time (s)	
GA	47.1	13.4	46.5	18	47	16.5	Random
PSO	47.1	17.5	46.5	14.8	47	15.5	
FA	47.1	35.5	46.5	53.8	47	38.8	
CSA	47.1	12.5	46.5	11.4	47	12.2	
GA	46.6	14.4	44	26.4	45.2	22.5	Latin Hypercube
PSO	46.6	16.8	44	24.9	45.2	28.7	
FA	46.6	33.5	44	54.2	45.2	32.5	
CSA	46.6	10.5	44	23	45.2	20.2	

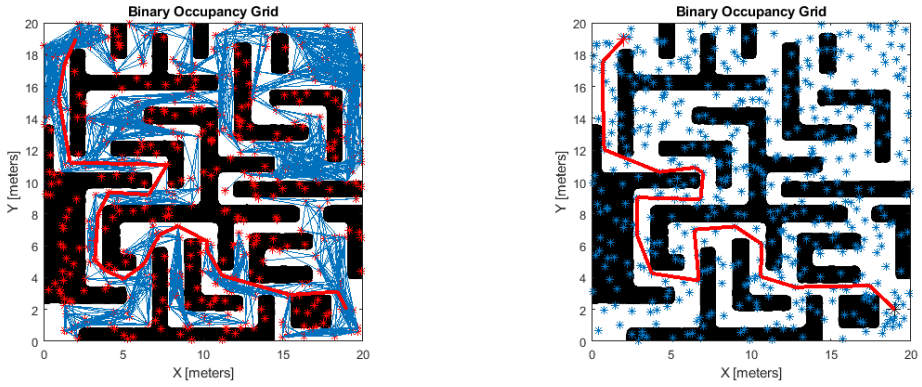


a



b

Figure 25: No-Free-Launch analysis of algorithms on 20 x 20 maze map (a) Comparison on path length (b) Comparison on time



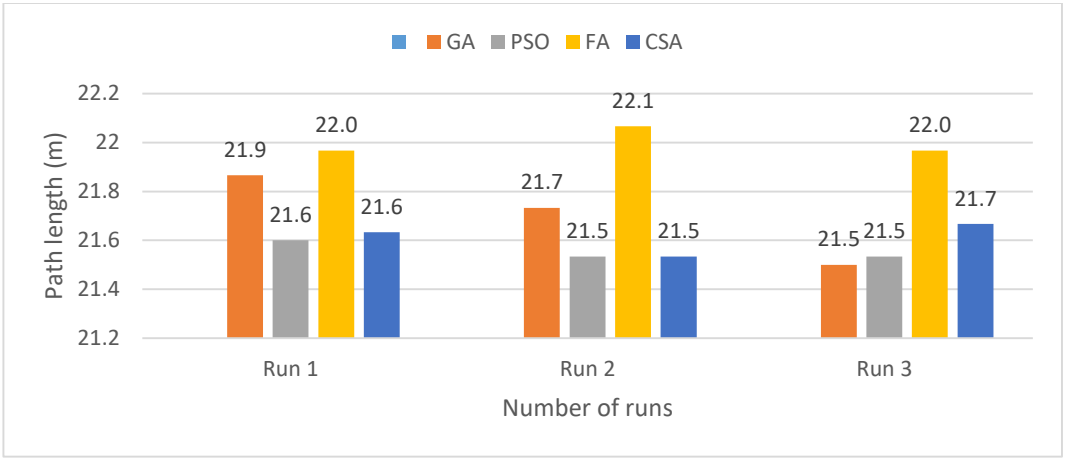
a b

Figure 26: Path planning on sampling methods: (a) Random sampling (b) Latin Hypercube sampling

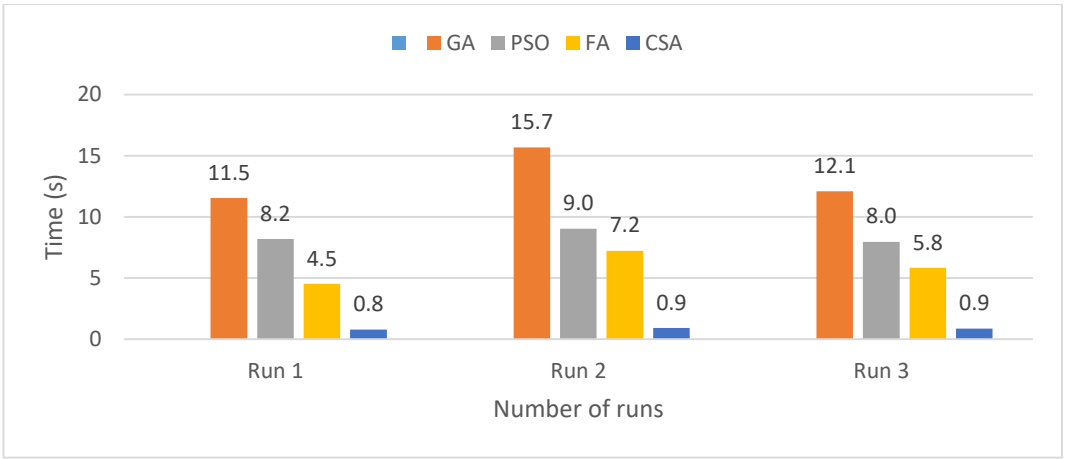
On a simple map like the 20 x 20 pothole map, PSO and CSA produced the shortest path in all three runs under random sampling, and PSO and GA achieved same outcome for all three runs under latin hypercube sampling as shown in Table 24. In applying the NFL theorem, the average of each algorithm across all sampling methods is obtained and the plotted bar chart is shown in Figure 27Figure 20. GA, when compared to FA gives a better result on path length, although FA finds the optimum oath in less time thus meeting the conditions of NFL theorem. Although CSA outperforms all algorithms in terms of optimization time, it obtained same shortest distance as PSO for run 1 and 2 and performed poorly against PSO in the third run, although the value didn't deviate much as compared to FA's result. But comparing the results through all three runs, CSA can be said to have performed better. The nature of paths generated with random sampling method on pothole map for all algorithms is shown in Figure 28.

Table 24: Analysis of algorithms on 20 x 20 pothole map

Algorithms	Run 1		Run 2		Run 3		Sampling
	Path length (m)	Time (s)	Path length (m)	Time (s)	Path length (m)	Time (s)	
GA	21.6	14.6	21.5	26.6	21.6	19.8	Random
PSO	21.6	7.9	21.5	13.8	21.6	11.2	
FA	21.6	1.4	21.6	2.0	21.7	3.3	
CSA	21.6	0.1	21.5	0.1	21.6	0.2	
GA	21.6	12.5	21.4	13.3	21.3	11.4	Latin Hypercube
PSO	21.6	10.8	21.4	7.4	21.3	8.2	
FA	21.8	1.8	21.6	3.8	21.4	3.0	
CSA	21.6	0.2	21.5	0.1	21.3	0.1	
GA	22.4	7.5	22.3	7.1	21.6	5.1	Pseudo-uniform
PSO	21.6	5.9	21.7	5.9	21.7	4.5	
FA	22.5	10.4	23.0	15.9	22.8	11.2	
CSA	21.7	2.1	21.6	2.6	22.1	2.3	



a



b

Figure 27: No-Free-Launch analysis of algorithms on 20 x 20 pothole map (a) Comparison on path length (b) Comparison on time

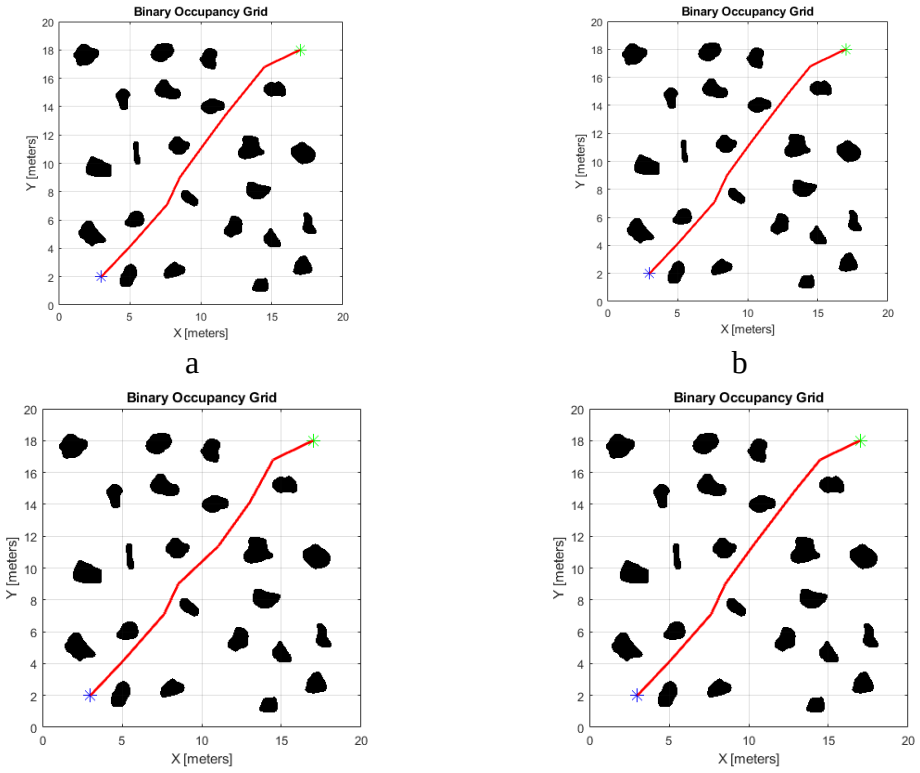


Figure 28: Random sampling on pothole map, run 3 (a) GA (b) PSO (c) FA (d) CSA

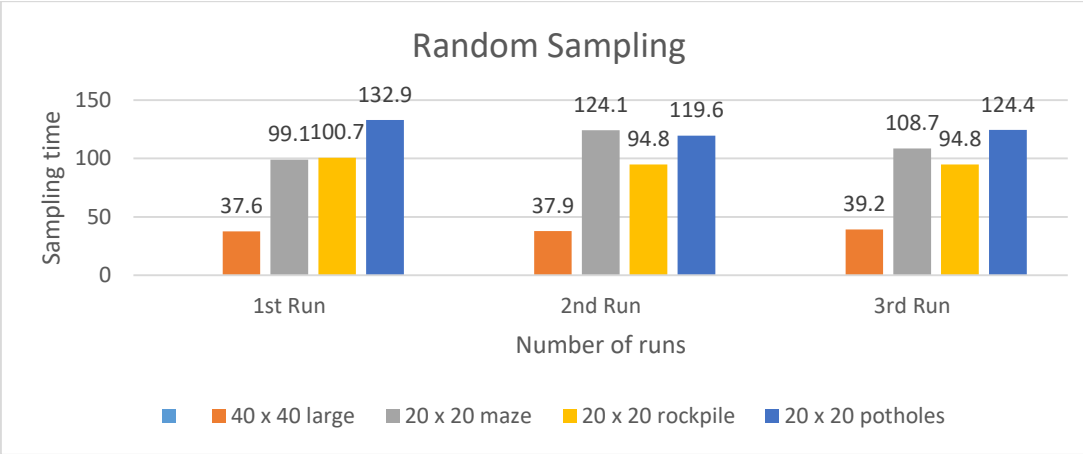
4.5.2 Comparison on Sampling Methods

As discussed earlier, the positions of samples on a map affect the nature of a planned path as shown in Figure 14. A comparison of the time it takes to sample points for different maps is tabulated on Table 25. As observed from Figure 29, all sampling methods take a longer time to distributes points on a pothole map. The reason is because the sampling process involves placing and connecting points together, therefore, because the obstacles on pothole maps are small, more points fall on the free space, thereby yielding an increased connection time.

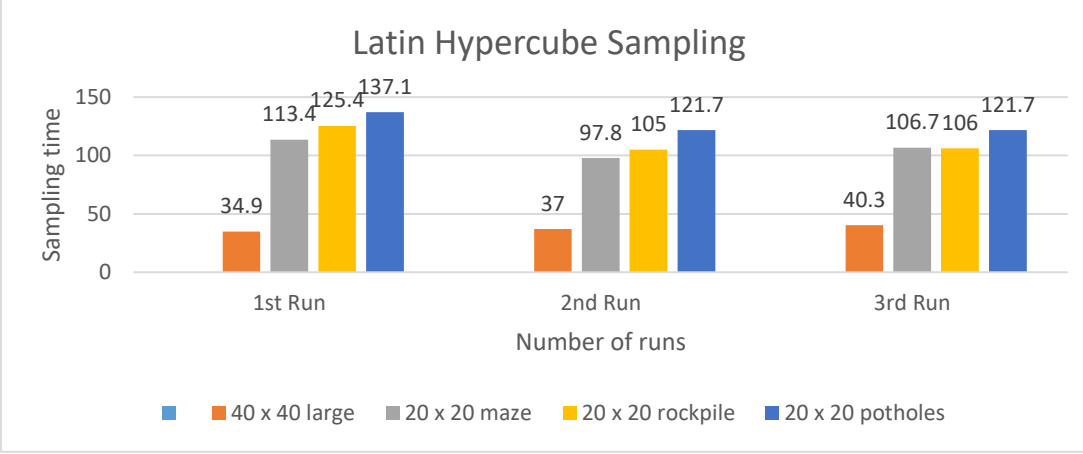
Table 25: Comparison on sampling time

Sampling method	Map	1st Run	2nd Run	3rd Run
Random Sampling	40 x 40 large	37.6	37.9	39.2
	20 x 20 maze	99.1	124.1	108.7
	20 x 20 rockpile	100.7	94.8	94.8

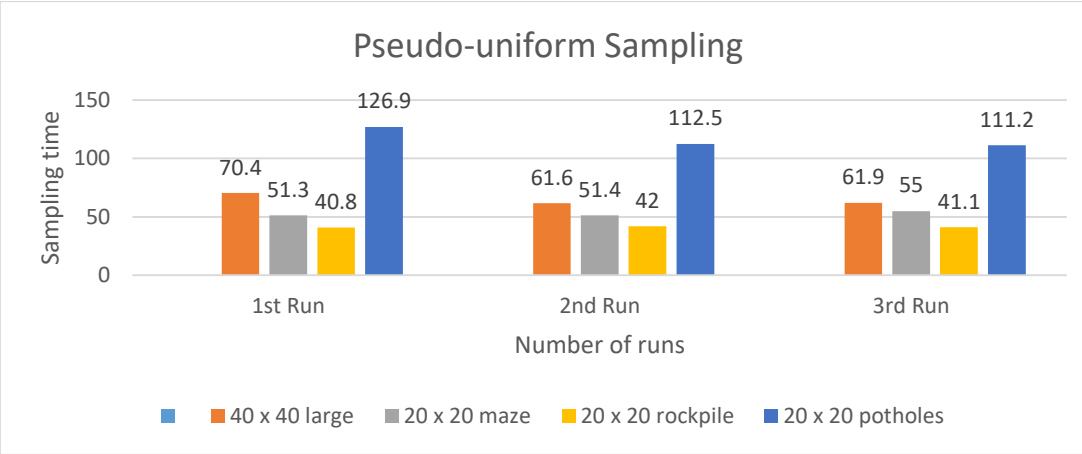
Sampling method	Map	1st Run	2nd Run	3rd Run
	20 x 20 potholes	132.9	119.6	124.4
Latin Hypercube Sampling	40 x 40 large	34.9	37.0	40.3
	20 x 20 maze	113.4	97.8	106.7
	20 x 20 rockpile	125.4	105.0	106.0
	20 x 20 potholes	137.1	121.7	121.7
Pseudo-uniform sampling	40 x 40 large	70.4	61.6	61.9
	20 x 20 maze	51.3	51.4	55.0
	20 x 20 rockpile	40.8	42.0	41.1
	20 x 20 potholes	126.9	112.5	111.2



a



b



c

Figure 29: Sampling time from different sampling methods (a) Random (b) Latin Hypercube (c) Pseudo-uniform.

There is a possibility of unsuccessful sampling of points on a map. Due to the nature of pseudo-uniform sampling method in distributing points at a particular angle, it fails to create a feasible solution space on maze map for a wide distance range because of the complexity of obstacles. This is shown in Figure 30.

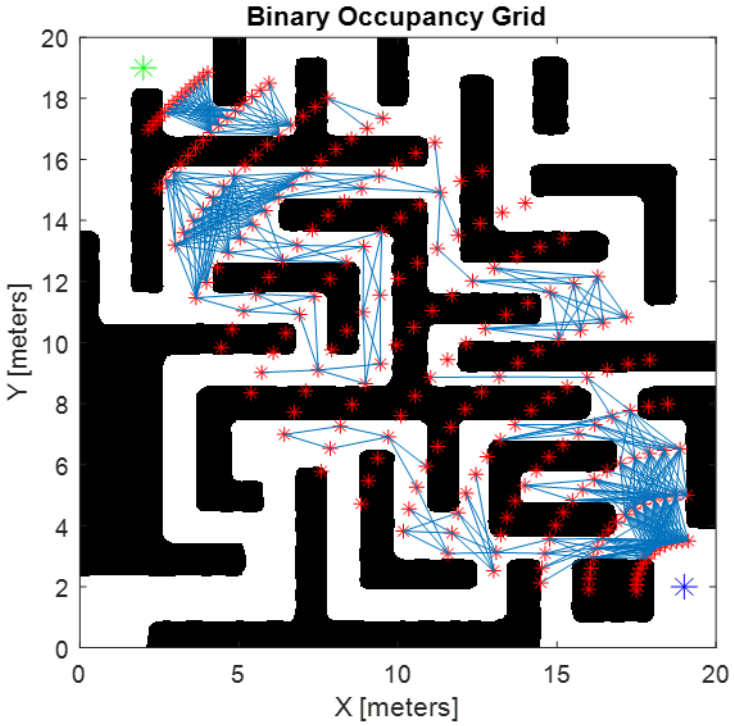


Figure 30: Unsuccessful pseudo-uniform sampling on maze map.

4.5.3 Comparison with Published Results

Analysis was carried out to compare with the results of Li et.al's work on path planning based on pseudo-sampling method [218]. Pseudo-uniform sampling method was utilized with the total sampling points ($N = m \times n$, where m and n denotes horizontal and vertical sampling points respectively) being 30, 60 and 90. Both path length and total composition time were considered and the test was recorded 10 times. Figure 31 and Table 26 shows the compared results in mean values. The start and goal points as estimated from the author's work are (405,43) and (47,405) respectively. This analysis was performed with the map (450 x 455 m) used in the author's work.

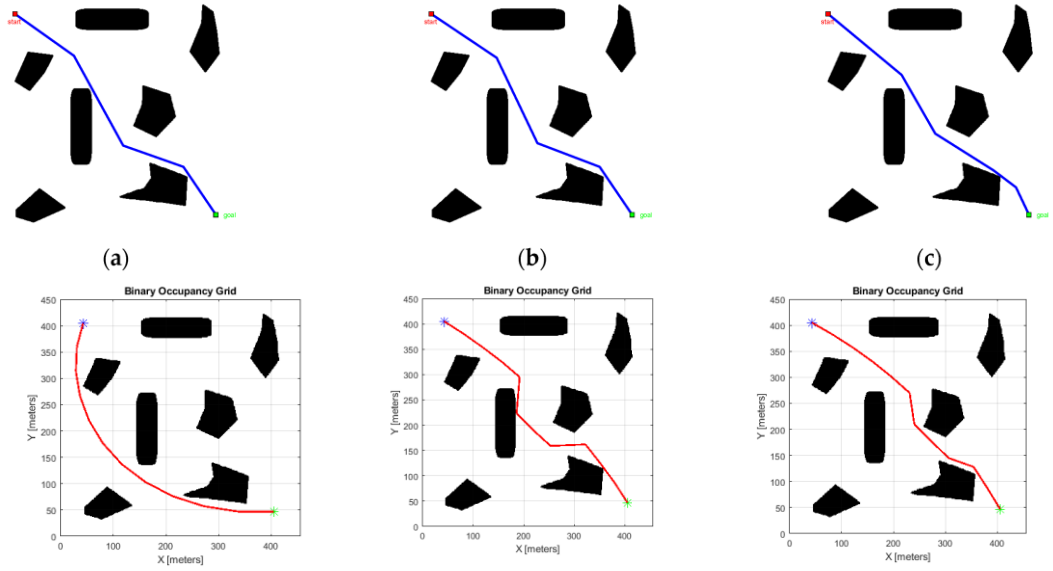


Figure 31: Comparison of results: (a) Li et. al, N = 30, (b) Li et. al, N = 60, (c) Li et. al, N = 90 (d) N = 30, (e) N = 60, (f) N = 90.

Table 26: Comparison on Path Length and Total Time

Number of Sampling points (N)	***Path length/m***					***Total time/s***				
	Li et. al	***GA***	***PSO***	***FA***	***CSA***	***Li***	***GA***	***PSO***	***FA***	***CSA***
30	593.3	605.1	605.1	605.1	605.1	0.404	6.332	6.041	3.539	4.852
60	590.6	561.4	561.4	561.4	561.4	2.056	12.075	12.840	14.343	11.980
90	578.7	532	532	532	532	5.196	23.676	24.015	28.016	22.798

It is observed from Table 26 that the results of the four algorithms are same. This is as a result of the algorithms optimizing based on few sampled points (30,60,90) on the map, so they all converge to same results. When the number of sample points is 30, there's no great advantage in the planned path compared to the author's result. This extends to the total construction time as the author's result shows less time. At N = 60, there is a 4.94% reduction in the path distance from the author's result, although that can't be said the in the construction time as there's a big difference. It can be observed that as the number of points increase, the time it takes to obtain a path increases as well. This is due to the fact that as the points increases, it takes more time to sample and connect them within the configuration free space. There's 8.07% reduction in path length when the number of points is 90. More sampled points give a greater chance of having a short path, and it also gives the algorithms the floor to show their difference in performance due to larger solution space.

The four algorithms gives a better result in path distance when N is 60 and 90 respectively, although there is a poor performance in the total planning time when compared to the authors result.

Chapter 5

CONCLUSION

In this study, path planning problem is treated as an optimization problem. Two important parameters are considered throughout this study for the path planning of smart vehicles: path distance and computation time. An extensive review of the various metaheuristic algorithms implemented for path planning problem is conducted and four algorithms namely CSA, GA, FA, and PSO are considered and utilized in this study. Four benchmark maps have been investigated in this study which are 40 x 40m large, 20 x 20m maze, 20 x 20m pothole, 20 x 20m rockpile. Furthermore, 3 sampling methods (pseudo-uniform, random, latin-hypercube sampling) are implemented to generate nodes on the maps which define paths for the algorithms. To optimize a path, a map is determined, a chosen sampling method samples points on the map, an initial path is determined in order to define the path size (number of points that make up a path), then a population is chosen for the algorithms to optimize.

For each map and each sampling method, the considered algorithms were analysed to determine the path distance and obtain the time it took to find an optimum path. Three runs were performed for each algorithm to obtain the optimal results. The performance of the algorithms was analysed based on each map, and subsequently each sampling methods. In majority of the runs, CSA obtained the shortest distance in least time as compared to other algorithms. Although CSA performed poorly on path distance using GA and PSO on the maze and pothole map respectively, its performance on other two

maps was relatively better both on shortest distance and less optimization time. Results also showed that sampling methods take a long time to distribute points on maps with very less complex obstacles (pothole map). However, from the results, it was observed that no single algorithm outperformed the rest on both shortest distance and minimum optimization time. Although, CSA had the least computational time for all cases, however, it is concluded that no single algorithm is universally the best-performing algorithm for all maps.

This study was further extended to compare results obtained in a published study. Although the author's result showed a better performance in obtaining the shortest path length when the number of sampled points was 30, that performance dropped as the sampled points increased to 60 and 90. There was a clear difference in the path length planning time as the author's result showed better performance in all three selected sampling points. All simulations were performed on R2022b MATLAB.

A very notable limitation to this work was the absence of some built-in algorithms in MATLAB. As of the time of this study, only GA and PSO metaheuristics are built-in libraries in MATLAB. Besides, the scope of this study covers the discrete nature of these algorithms. Because the built-in algorithms work with continuous data (except for GA which can optimize both discrete and continuous data), the discrete metaheuristics were obtained from MathWork's FileExchange website and some were built out of some researcher's work on discrete metaheuristic algorithms as cited in section 3.3. This study can further be extended in the future to include path planning on 3D maps. Also, analysis with dynamic obstacles can be considered in the optimization of path distance in navigating smart vehicles.

REFERENCES

[1] Shi Y, Han Q, Shen W, Wang X. A Multi-Layer Collaboration Framework for Industrial Parks with 5G Vehicle-to-Everything Networks. Engineering 2021. https://doi.org/10.1016/j.eng.2020.12.021.

[2] Gasmi R, Harous S. Robust Connectivity-Based Internet of Vehicles Clustering Algorithm. Wirel Pers Commun 2022;125:3153–85. https://doi.org/10.1007/S11277-022-09703-0.

[3] Dadvandipour S, Ganie AG. An Approach to Implementation of Autoencoders in Intelligent Vehicles 2023:3–10. https://doi.org/10.1007/978-3-031-15211-5_1.

[4] Madhav AVS, Tyagi AK. Explainable Artificial Intelligence (XAI): Connecting Artificial Decision-Making and Human Trust in Autonomous Vehicles. Lect Notes Networks Syst 2023;421:123–36. https://doi.org/10.1007/978-981-19-1142-2_10.

[5] Theodosc R, Denis D, Blanc C, Chateau T, Checchin P. Vehicle detection based on deep learning heatmap estimation. IEEE Intell. Veh. Symp. Proc., 2019. https://doi.org/10.1109/IVS.2019.8814285.

[6] Thadeshwar H, Shah V, Jain M, Chaudhari R, Badgujar V. Artificial Intelligence based Self-Driving Car. 4th Int. Conf. Comput. Commun. Signal Process. ICCCSP 2020, 2020.

https://doi.org/10.1109/ICCCSP49186.2020.9315223.

[7] Ciuciu C, Ţă DBĂRBU, Ujan SSĂSĂ, Ş EŞIPO. Autonomous Scale Model Car with Ultrasonic Sensors and Arduino Board. Acta Tech Napocensis 2017.

[8] Zhang Z, Chen J, Guo Q. Application of Automated Guided Vehicles in Smart Automated Warehouse Systems: A Survey. Comput Model Eng Sci 2023;134:1529–63. https://doi.org/10.32604/CMES.2022.021451.

[9] Anvo R, Kaur A, Sattar TP. Wireless Communication with Mobile Inspection Robots Operating While Submerged Inside Oil Storage Tanks. Lect. Notes Networks Syst., 2022. https://doi.org/10.1007/978-3-030-86294-7_14.

[10] Coandă P, Avram M, Constantin V, Grămescu B. Indoor GPS System for Autonomous Mobile Robots Used in Surveillance Applications. Lect. Notes Mech. Eng., 2022. https://doi.org/10.1007/978-3-030-79168-1_9.

[11] Grămescu B, Cartal A, Hashim AS, Nițu C. Dynamic Analysis of a Robot Locomotion for an External Pipe Inspection and Monitoring. Lect. Notes Mech. Eng., 2022. https://doi.org/10.1007/978-3-030-79168-1_18.

[12] Yıldırım Ş, Savaş S. Design of a Mobile Robot to Work in Hospitals and Trajectory Planning Using Proposed Neural Networks Predictors. Lect. Notes Networks Syst., 2022. https://doi.org/10.1007/978-3-030-83368-8_4.

[13] Zhou Z, Li L, Fürsterling A, Durocher HJ, Mouridsen J, Zhang X. Learning-

based object detection and localization for a mobile robot manipulator in SME production. Robot Comput Integr Manuf 2022. https://doi.org/10.1016/j.rcim.2021.102229.

[14] Dellmann T, Berns K. Toward a Realistic Simulation for Agricultural Robots. Smart Innov. Syst. Technol., 2022. https://doi.org/10.1007/978-981-16-3349-2_1.

[15] Galati R, Mantriota G, Reina G. Mobile Robotics for Sustainable Development: Two Case Studies, 2022. https://doi.org/10.1007/978-3-030-87383-7_41.

[16] Fu J, Tian F, Chai T, Jing Y, Li Z, Su CY. Motion Tracking Control Design for a Class of Nonholonomic Mobile Robot Systems. IEEE Trans Syst Man, Cybern Syst 2020;50. https://doi.org/10.1109/TSMC.2018.2804948.

[17] Mehrez MW, Worthmann K, Cenerini JPV, Osman M, Melek WW, Jeon S. Model Predictive Control without terminal constraints or costs for holonomic mobile robots. Rob Auton Syst 2020;127. https://doi.org/10.1016/j.robot.2020.103468.

[18] Holmberg R, Khatib O. Development and control of a holonomic mobile robot for mobile manipulation tasks. Int J Rob Res 2000. https://doi.org/10.1177/02783640022067977.

[19] Laumond JP, Jacobs PE, Taïx M, Murray RM. A Motion Planner for Nonholonomic Mobile Robots. IEEE Trans Robot Autom 1994.

https://doi.org/10.1109/70.326564.

[20] Koubaa A, Bennaceur H, Chaari I, Trigui S, Ammar A, Sriti MF, et al. Introduction to mobile robot path planning. Stud. Comput. Intell., 2018. https://doi.org/10.1007/978-3-319-77042-0_1.

[21] Mac TT, Copot C, Tran DT, De Keyser R. Heuristic approaches in robot path planning: A survey. Rob Auton Syst 2016. https://doi.org/10.1016/j.robot.2016.08.001.

[22] Wahab MNA, Nefti-Meziani S, Atyabi A. A comparative review on mobile robot path planning: Classical or meta-heuristic methods? Annu Rev Control 2020;50:233–52. https://doi.org/10.1016/J.ARCONTROL.2020.10.001.

[23] Alam MS, Rafique MU, Khan MU, Algorithm I. Mobile Robot Path Planning in Static Environments using Particle Swarm Optimization. Int J Comput Sci Electron Eng 2015;3.

[24] Karur K, Sharma N, Dharmatti C, Siegel JE. A Survey of Path Planning Algorithms for Mobile Robots. Vehicles 2021. https://doi.org/10.3390/vehicles3030027.

[25] Verma P, Alam A, Sarwar A, Tariq M, Vahedi H, Gupta D, et al. Meta-heuristic optimization techniques used for maximum power point tracking in solar pv system. Electron 2021. https://doi.org/10.3390/electronics10192419.

[26] Campbell S, O'Mahony N, Carvalho A, Krpalkova L, Riordan D, Walsh J. Path Planning Techniques for Mobile Robots A Review. 2020 6th Int. Conf. Mechatronics Robot. Eng. ICMRE 2020, 2020. https://doi.org/10.1109/ICMRE49073.2020.9065187.

[27] Blum C, Roli A. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Comput Surv 2003. https://doi.org/10.1145/937503.937505.

[28] Hans J. Bremermann. The Evolution of Intelligence: The Nervous System as a Model of Its Environment. University of Washington, Department of Mathematics; 1958.

[29] Holland JH. Adaptation in natural and artificial systems, University of Michigan press. Ann Arbor, MI 1975.

[30] Patle BK, Babu L G, Pandey A, Parhi DRK, Jagadeesh A. A review: On path planning strategies for navigation of mobile robot. Def Technol 2019;15. https://doi.org/10.1016/j.dt.2019.04.011.

[31] Pratihar DK, Deb K, Ghosh A. Fuzzy-genetic algorithms and time-optimal obstacle-free path generation for mobile robots. Eng Optim 1999. https://doi.org/10.1080/03052159908941294.

[32] Kumar S, Parhi DR, Pandey KK, Muni MK. Hybrid IWD-GA: An approach for path optimization and control of multiple mobile robot in obscure static and

dynamic environments. Robotica 2021. https://doi.org/10.1017/S0263574721000114.

[33] Hui NB, Pratihar DK. A comparative study on some navigation schemes of a real robot tackling moving obstacles. Robot Comput Integr Manuf 2009. https://doi.org/10.1016/j.rcim.2008.12.003.

[34] Fries TP. Evolutionary robot navigation using fuzzy terrain conditions. Annu. Conf. North Am. Fuzzy Inf. Process. Soc. - NAFIPS, 2006. https://doi.org/10.1109/NAFIPS.2006.365466.

[35] Hu Y, Yang SX. A knowledge based genetic algorithm for path planning of a mobile robot. Proc. - IEEE Int. Conf. Robot. Autom., 2004.

[36] Lei L, Wang H, Wu Q. Improved genetic algorithms based path planning of mobile robot under dynamic unknown environment. 2006 IEEE Int. Conf. Mechatronics Autom. ICMA 2006, 2006. https://doi.org/10.1109/ICMA.2006.257475.

[37] Lu J, Yang D. Path planning based on double-layer genetic algorithm. Proc. - Third Int. Conf. Nat. Comput. ICNC 2007, 2007. https://doi.org/10.1109/ICNC.2007.546.

[38] Mahjoubi H, Bahrami F, Lucas C. Path planning in an environment with static and dynamic obstacles using genetic algorithm: A simplified search space approach. 2006 IEEE Congr. Evol. Comput. CEC 2006, 2006.

https://doi.org/10.1109/cec.2006.1688617.

[39] Yuan J, Yu T, Wang K, Liu X. Step-spreading map knowledge based multi-objective genetic algorithm for robot-path planning. Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern., 2007. https://doi.org/10.1109/ICSMC.2007.4413953.

[40] Liu ML, Yao XZ, Huang JY, Zhang C. OPTIMIZATION OF UNMANNED VEHICLE SCHEDULING AND ORDER ALLOCATION. Int J Simul Model 2022;21:477–88. https://doi.org/10.2507/IJSIMM21-3-613.

[41] Wang WC, Ng CY, Chen R. Vision-aided path planning using low-cost gene encoding for a mobile robot. Intell Autom Soft Comput 2022. https://doi.org/10.32604/iasc.2022.022067.

[42] Alliez P, Fabri A. CGAL-The computational geometry algorithms library. ACM SIGGRAPH 2016 Courses, SIGGRAPH 2016, 2016. https://doi.org/10.1145/2897826.2927362.

[43] Naderan-Tahan M, Manzuri-Shalmani T. Efficient and safe path planning for a Mobile robot using genetic algorithm. 2009 IEEE Congr. Evol. Comput. CEC 2009, 2009. https://doi.org/10.1109/CEC.2009.4983199.

[44] Cheng KP, Mohan RE, Khanh Nhan NH, Le AV. Multi-Objective Genetic Algorithm-Based Autonomous Path Planning for Hinged-Tetro Reconfigurable Tiling Robot. IEEE Access 2020. https://doi.org/10.1109/ACCESS.2020.3006579.

[45] Fu Y. Path planning method of smart mobile robot based on genetic algorithm. J Phys Conf Ser 2021. https://doi.org/10.1088/1742-6596/2083/4/042014.

[46] M. D. Optimization, Learning and Natural Algorithms. PhD Thesis, Politec Di Milano 1992.

[47] Goss S, Aron S, Deneubourg JL, Pasteels JM. Self-organized shortcuts in the Argentine ant. Naturwissenschaften 1989. https://doi.org/10.1007/BF00462870.

[48] Dorigo M, Blum C. Ant colony optimization theory: A survey. Theor Comput Sci 2005. https://doi.org/10.1016/j.tcs.2005.05.020.

[49] Brand M, Masuda M, Wehner N, Yu XH. Ant colony optimization algorithm for robot path planning. 2010 Int. Conf. Comput. Des. Appl. ICCDA 2010, 2010. https://doi.org/10.1109/ICCDA.2010.5541300.

[50] Liu J, Yang J, Liu H, Tian X, Gao M. An improved ant colony algorithm for robot path planning. Soft Comput 2017. https://doi.org/10.1007/s00500-016-2161-7.

[51] You XM, Liu S, Zhang C. An improved ant colony system algorithm for robot path planning and performance analysis. Int J Robot Autom 2018. https://doi.org/10.2316/Journal.206.2018.5.206-0071.

[52] Dai X, Long S, Zhang Z, Gong D. Mobile robot path planning based on ant

colony algorithm with a∗ heuristic method. Front Neurorobot 2019. https://doi.org/10.3389/fnbot.2019.00015.

[53] 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. https://doi.org/10.1016/j.jocs.2018.02.004.

[54] Akka K, Khaber F. Mobile robot path planning using an improved ant colony optimization. Int J Adv Robot Syst 2018. https://doi.org/10.1177/1729881418774673.

[55] Yang L, Fu L, Li P, Mao J, Guo N, Du L. LF-ACO: An effective formation path planning for multi-mobile robot. Math Biosci Eng 2022. https://doi.org/10.3934/mbe.2022012.

[56] Yang H, Qi J, Miao Y, Sun H, Li J. A New Robot Navigation Algorithm Based on a Double-Layer Ant Algorithm and Trajectory Optimization. IEEE Trans Ind Electron 2019. https://doi.org/10.1109/TIE.2018.2886798.

[57] Zhao J, Cheng D, Hao C. An Improved Ant Colony Algorithm for Solving the Path Planning Problem of the Omnidirectional Mobile Vehicle. Math Probl Eng 2016. https://doi.org/10.1155/2016/7672839.

[58] Tao Y, Gao H, Ren F, Chen C, Wang T, Xiong H, et al. A mobile service robot global path planning method based on ant colony optimization and fuzzy control. Appl Sci 2021. https://doi.org/10.3390/app11083605.

[59] Zhang D, Luo R, Yin Y, Zou S. Multi-objective path planning for mobile robot in nuclear accident environment based on improved ant colony optimization with modified A*. Nucl Eng Technol 2023. https://doi.org/10.1016/J.NET.2023.02.005.

[60] Miao C, Chen G, Yan C, Wu Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. Comput Ind Eng 2021;156. https://doi.org/10.1016/j.cie.2021.107230.

[61] Eberhart, R., & Kennedy J(. Particle Swamp Optimization. IEEE Int. Conf. neural networks, 1995.

[62] Rendón MA, Martins FF. Path Following Control Tuning for an Autonomous Unmanned Quadrotor Using Particle Swarm Optimization. IFAC-PapersOnLine, 2017. https://doi.org/10.1016/j.ifacol.2017.08.054.

[63] Kumar PB, Pandey KK, Sahu C, Chhotray A, Parhi DR. A hybridized RA-APSO approach for humanoid navigation. 2017 Nirma Univ. Int. Conf. Eng. NUiCONE 2017, 2018. https://doi.org/10.1109/NUICONE.2017.8325611.

[64] Gao M, Ding P, Yang Y. Time-optimal trajectory planning of industrial robots based on particle swarm optimization. Proc. - 5th Int. Conf. Instrum. Meas. Comput. Commun. Control. IMCCC 2015, 2016. https://doi.org/10.1109/IMCCC.2015.410.

[65] Aydilek IB, Nacar MA, GüMüŞÇü A, Salur MU. Comparing inertia weights of

particle swarm optimization in multimodal functions. IDAP 2017 - Int. Artif. Intell. Data Process. Symp., 2017. https://doi.org/10.1109/IDAP.2017.8090225.

[66] Raska P, Ulrych Z. Testing different particle swarm optimization strategies. Proc. 30th Int. Bus. Inf. Manag. Assoc. Conf. IBIMA 2017 - Vis. 2020 Sustain. Econ. Dev. Innov. Manag. Glob. Growth, 2017.

[67] Dai HP, Chen DD, Zheng ZS. Effects of random values for particle swarm optimization algorithm. Algorithms 2018. https://doi.org/10.3390/A11020023.

[68] Shi Y, Eberhart RC. Parameter selection in particle swarm optimization BT - Evolutionary Programming VII. 1998 Int Conf Evol Program 1998.

[69] Ratnaweera A, Halgamuge SK, Watson HC. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 2004. https://doi.org/10.1109/TEVC.2004.826071.

[70] Suryanto N, Ikuta C, Pramadihanto D. Multi-group particle swarm optimization with random redistribution. Proc. - Int. Electron. Symp. Knowl. Creat. Intell. Comput. IES-KCIC 2017, 2017. https://doi.org/10.1109/KCIC.2017.8228445.

[71] Dewang HS, Mohanty PK, Kundu S. A Robust Path Planning For Mobile Robot Using Smart Particle Swarm Optimization. Procedia Comput Sci 2018;133:290–7. https://doi.org/10.1016/J.PROCS.2018.07.036.

[72] Chai Q, Wang Y, He Y, Xu C, Hong Z. Improved PRM Path Planning in Narrow Passages Based on PSO. 2022 IEEE Int Conf Mechatronics Autom ICMA 2022 2022:41–6. https://doi.org/10.1109/ICMA54519.2022.9855913.

[73] Masehian E, Sedighizadeh D. A multi-objective PSO-based algorithm for robot path planning. Proc. IEEE Int. Conf. Ind. Technol., 2010. https://doi.org/10.1109/ICIT.2010.5472755.

[74] Li X, Wu D, He J, Bashir M, Liping M. An Improved Method of Particle Swarm Optimization for Path Planning of Mobile Robot. J Control Sci Eng 2020. https://doi.org/10.1155/2020/3857894.

[75] Song B, Wang Z, Zou L. An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve. Appl Soft Comput 2021;100. https://doi.org/10.1016/j.asoc.2020.106960.

[76] Amar LB, Jasim WM. Hybrid metaheuristic approach for robot path planning in dynamic environment. Bull Electr Eng Informatics 2021. https://doi.org/10.11591/EEI.V10I4.2836.

[77] Fernandes PB, Oliveira RCL, Fonseca Neto JV. Trajectory planning of autonomous mobile robots applying a particle swarm optimization algorithm with peaks of diversity. Appl Soft Comput 2022;116:108108. https://doi.org/10.1016/J.ASOC.2021.108108.

[78] Xu L, Cao M, Song B. A new approach to smooth path planning of mobile robot

based on quartic Bezier transition curve and improved PSO algorithm. Neurocomputing 2022;473:98–106. https://doi.org/10.1016/J.NEUCOM.2021.12.016.

[79] Han Y, Zhang L, Tan H, Xue X. Mobile robot path planning based on improved particle swarm optimization. Chinese Control Conf CCC 2019;2019-July:4354–8. https://doi.org/10.23919/CHICC.2019.8866634.

[80] Sathiya V, Chinnadurai M, Ramabalan S. Mobile robot path planning using fuzzy enhanced improved Multi-Objective particle swarm optimization (FIMOPSO). Expert Syst Appl 2022;198:116875. https://doi.org/10.1016/J.ESWA.2022.116875.

[81] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. J Glob Optim 2007. https://doi.org/10.1007/s10898-007-9149-x.

[82] Contreras-Cruz MA, Ayala-Ramirez V, Hernandez-Belmonte UH. Mobile robot path planning using artificial bee colony and evolutionary programming. Appl Soft Comput 2015. https://doi.org/10.1016/j.asoc.2015.01.067.

[83] Faridi AQ, Sharma S, Shukla A, Tiwari R, Dhar J. Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment. Intell Serv Robot 2018. https://doi.org/10.1007/s11370-017-0244-7.

[84] Kumar S, Sikander A. Optimum Mobile Robot Path Planning Using Improved Artificial Bee Colony Algorithm and Evolutionary Programming. Arab J Sci Eng 2022. https://doi.org/10.1007/s13369-021-06326-8.

[85] Kamil R, Mohamed M, Oleiwi B. Path Planning of Mobile Robot Using Improved Artificial Bee Colony Algorithm. Eng Technol J 2020. https://doi.org/10.30684/etj.v38i9a.1100.

[86] HadiAbbas N, Mahdi Ali F. Path Planning of an Autonomous Mobile Robot using Directed Artificial Bee Colony Algorithm. Int J Comput Appl 2014. https://doi.org/10.5120/16836-6681.

[87] Szczepanski R, Tarczewski T. Global path planning for mobile robot based on Artificial Bee Colony and Dijkstra's algorithms. Proc. - 2021 IEEE 19th Int. Power Electron. Motion Control Conf. PEMC 2021, 2021. https://doi.org/10.1109/PEMC48073.2021.9432570.

[88] Sudhakara P, Ganapathy V, Sundaran K. Mobile robot trajectory planning using enhanced artificial bee colony optimization algorithm. IEEE Int. Conf. Power, Control. Signals Instrum. Eng. ICPCSI 2017, 2018. https://doi.org/10.1109/ICPCSI.2017.8392316.

[89] Yang X. Nature-Inspired Metaheuristic Algorithms. 2010.

[90] Qi X, Zhu S, Zhang H. A hybrid firefly algorithm. Proc. 2017 IEEE 2nd Adv. Inf. Technol. Electron. Autom. Control Conf. IAEAC 2017, 2017.

https://doi.org/10.1109/IAEAC.2017.8054023.

[91] Ali SM, Yonan JF, Alniemi O, Ahmed AA. Mobile Robot Path Planning Optimization Based on Integration of Firefly Algorithm and Cubic Polynomial Equation. J ICT Res Appl 2022;16:1–22. https://doi.org/10.5614/ITBJ.ICT.RES.APPL.2022.16.1.1.

[92] Kundra H, Khan W, Malik M, Rane KP, Neware R, Jain V. Quantum-inspired firefly algorithm integrated with cuckoo search for optimal path planning. Int J Mod Phys C 2022;33. https://doi.org/10.1142/S0129183122500188.

[93] Mohammadi M, Mobarakeh MI. An integrated clustering algorithm based on firefly algorithm and self-organized neural network. Prog Artif Intell 2022;11:207–17. https://doi.org/10.1007/S13748-022-00275-5.

[94] Zivkovic M, Petrovic A, Venkatachalam K, Strumberger I, Jassim HS, Bacanin N. Novel Chaotic Best Firefly Algorithm: COVID-19 Fake News Detection Application. Stud Comput Intell 2023;1054:285–305. https://doi.org/10.1007/978-3-031-09835-2_16.

[95] Moharamkhani E, Yahyaei Feriz Hendi M, Bandar E, Izadkhasti A, Sirwan Raza R. Intrusion detection system based firefly algorithm-random forest for cloud computing. Concurr Comput Pract Exp 2022. https://doi.org/10.1002/CPE.7220.

[96] Chen X, Zhou M, Huang J, Luo Z. Global path planning using modified firefly

algorithm. MHS 2017 - 28th 2017 Int. Symp. Micro-NanoMechatronics Hum. Sci., vol. 2018- January, 2018. https://doi.org/10.1109/MHS.2017.8305195.

[97] Duan P, Li J, Sang H, Han Y, Sun Q. A Developed Firefly Algorithm for Multi-Objective Path Planning Optimization Problem. 8th Annu. IEEE Int. Conf. Cyber Technol. Autom. Control Intell. Syst. CYBER 2018, 2019. https://doi.org/10.1109/CYBER.2018.8688342.

[98] Patle BK, Pandey A, Jagadeesh A, Parhi DR. Path planning in uncertain environment by using firefly algorithm. Def Technol 2018;14:691–701. https://doi.org/10.1016/J.DT.2018.06.004.

[99] Abdul Hassan AK, Fadhil DJ. Mobile Robot Path Planning Method Using Firefly Algorithm for 3D Sphere Dynamic & Partially Known Environment. J Univ BABYLON Pure Appl Sci 2018;26. https://doi.org/10.29196/jubpas.v26i7.1506.

[100] Zghair NAK, Al-Araji AS. Intelligent Hybrid Path Planning Algorithms for Autonomous Mobile Robots. Int J Intell Eng Syst 2022;15:309–25. https://doi.org/10.22266/IJIES2022.1031.28.

[101] Hidalgo-Paniagua A, Vega-Rodríguez MA, Ferruz J, Pavón N. Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. Soft Comput 2017;21:949–64. https://doi.org/10.1007/S00500-015-1825-Z/FIGURES/21.

[102] Singh NH, Laishram A, Thongam K. Optimal Path Planning for Mobile Robot Navigation Using FA-TPM in Cluttered Dynamic Environments. Procedia Comput Sci 2023;218:612–20. https://doi.org/10.1016/J.PROCS.2023.01.043.

[103] Yang XS, Deb S. Cuckoo search via Lévy flights. 2009 World Congr. Nat. Biol. Inspired Comput. NABIC 2009 - Proc., 2009. https://doi.org/10.1109/NABIC.2009.5393690.

[104] Joshi AS, Kulkarni O, Kakandikar GM, Nandedkar VM. Cuckoo Search Optimization- A Review. Mater. Today Proc., 2017. https://doi.org/10.1016/j.matpr.2017.07.055.

[105] Valian E, Mohanna S, Tavakoli S. Improved Cuckoo Search Algorithm for Global Optimization. Int J Commun Inf Technol 2011.

[106] Xiao L, Hajjam-El-Hassani A, Dridi M. An application of extended cuckoo search to vehicle routing problem. 2017 Int. Colloq. Logist. Supply Chain Manag. Compet. Innov. Automob. Aeronaut. Ind. LOGISTIQUA 2017, 2017. https://doi.org/10.1109/LOGISTIQUA.2017.7962869.

[107] Karagul K, Sahin Y. An Improved Cuckoo Search Algorithm for the Capacitated Green Vehicle Routing Problem. Stud Comput Intell 2023;1054:385–406. https://doi.org/10.1007/978-3-031-09835-2_21.

[108] Chatterjee S, Dey N, Sen S, Ashour AS, Fong SJ, Shi F. Modified cuckoo search ased neural networks for forest types classification. Front. Artif. Intell. Appl.,

2017. https://doi.org/10.3233/978-1-61499-785-6-490.

[109] Bibiks K, Hu YF, Li JP, Pillai P, Smith A. Improved discrete cuckoo search for the resource-constrained project scheduling problem. Appl Soft Comput J 2018. https://doi.org/10.1016/j.asoc.2018.04.047.

[110] Prakash Tiwari S, Singh G. Optimizing Job Scheduling Problem Using Improved GA + CS Algorithm 2023:291–7. https://doi.org/10.1007/978-981-19-2821-5_25.

[111] Shehab M, Khader AT, Al-Betar MA. A survey on applications and variants of the cuckoo search algorithm. Appl Soft Comput J 2017. https://doi.org/10.1016/j.asoc.2017.02.034.

[112] Lakshmi SA, Anandavelu K. Enhanced Cuckoo Search Optimization Technique for Skin Cancer Diagnosis Application. Intell Autom Soft Comput 2023;35:3403–13. https://doi.org/10.32604/IASC.2023.030970.

[113] Parkash D, Mittal S. An Enhanced Secure Framework Using CSA for Cloud Computing Environments 2023:349–56. https://doi.org/10.1007/978-981-19-2535-1_27.

[114] Sahu B, Kumar Das P, Ranjan Kabat M. Multi-robot Cooperation and Path Planning Using Modified Cuckoo Search 2023:369–82. https://doi.org/10.1007/978-981-19-1412-6_31.

[115] Garip Z, Karayel D, Erhan Çimen M. A study on path planning optimization of mobile robots based on hybrid algorithm. Concurr Comput Pract Exp 2022;34. https://doi.org/10.1002/CPE.6721.

[116] Kumar S, Parhi DRK, Kashyap AK, Vikas. Path Optimization and Control of Mobile Robot Using Modified Cuckoo Search Algorithm. Lect Notes Mech Eng 2022:125–33. https://doi.org/10.1007/978-981-19-0296-3_12.

[117] Sahu B, Das PK, Kumar R. A modified cuckoo search algorithm implemented with SCA and PSO for multi-robot cooperation and path planning. Cogn Syst Res 2023;79:24–42. https://doi.org/10.1016/J.COGSYS.2023.01.005.

[118] Fan Y, Sun X, Wang G, Mu D. Collision avoidance controller for unmanned surface vehicle based on improved cuckoo search algorithm. Appl Sci 2021;11. https://doi.org/10.3390/APP11209741.

[119] Saraswathi M, Murali GB, Deepak BBVL. Optimal Path Planning of Mobile Robot Using Hybrid Cuckoo Search-Bat Algorithm. Procedia Comput. Sci., 2018. https://doi.org/10.1016/j.procs.2018.07.064.

[120] Wang J, Shang X, Guo T, Zhou J, Jia S, Wang C. Optimal path planning based on hybrid genetic-cuckoo search algorithm. 2019 6th Int. Conf. Syst. Informatics, ICSAI 2019, 2019. https://doi.org/10.1109/ICSAI48974.2019.9010519.

[121] Gunji B, Deepak BBVL, Saraswathi MBL, Mogili UR. Optimal path planning

of mobile robot using the hybrid cuckoo–bat algorithm in assorted environment. Int J Intell Unmanned Syst 2019. https://doi.org/10.1108/IJIUS-07-2018-0021.

[122] Mirjalili S, Lewis A. The Whale Optimization Algorithm. Adv Eng Softw 2016. https://doi.org/10.1016/j.advengsoft.2016.01.008.

[123] Watkins WA, Schevill WE. Aerial Observation of Feeding Behavior in Four Baleen Whales: Eubalaena glacialis, Balaenoptera borealis, Megaptera novaeangliae, and Balaenoptera physalus. J Mammal 1979. https://doi.org/10.2307/1379766.

[124] Abdel-Basset M, Mohamed R, Abouhawwash M. A new fusion of whale optimizer algorithm with Kapur's entropy for multi-threshold image segmentation: analysis and validations. Artif Intell Rev 2022. https://doi.org/10.1007/s10462-022-10157-w.

[125] Taheri AA, Golabi S. Thickness Optimization and Experimental Validation of Incremental Collar Forming of Explosive Welded Al/Cu Bimetal Sheet with an Obround Hole Using Finite Element, Whale Algorithm and Neural Network. Iran J Sci Technol - Trans Mech Eng 2022. https://doi.org/10.1007/s40997-021-00445-1.

[126] Lakshmi AV, Mohanaiah P. Intelligent facial emotion recognition based on Hybrid whale optimization algorithm and sine cosine algorithm. Microprocess Microsyst 2022;95:104718. https://doi.org/10.1016/J.MICPRO.2022.104718.

[127] Butti D, Mangipudi SK, Rayapudi SR. An improved whale optimization algorithm for the design of multi-machine power system stabilizer. Int Trans Electr Energy Syst 2020. https://doi.org/10.1002/2050-7038.12314.

[128] Abdel-Basset M, El-Shahat D, Deb K, Abouhawwash M. Energy-aware whale optimization algorithm for real-time task scheduling in multiprocessor systems. Appl Soft Comput J 2020. https://doi.org/10.1016/j.asoc.2020.106349.

[129] Li X, Yang Q, Wu H, Tan S, He Q, Wang N, et al. Joints Trajectory Planning of Robot Based on Slime Mould Whale Optimization Algorithm. Algorithms 2022;15. https://doi.org/10.3390/A15100363.

[130] Li G, Cui Y, Wang L, Meng L. Automatic Registration Algorithm for the Point Clouds Based on the Optimized RANSAC and IWOA Algorithms for Robotic Manufacturing. Appl Sci 2022;12. https://doi.org/10.3390/APP12199461.

[131] Zong X, Liu J, Ye Z, Liu Y. Whale optimization algorithm based on Levy flight and memory for static smooth path planning. Int J Mod Phys C 2022;33. https://doi.org/10.1142/S0129183122501388.

[132] Gul F, Mir S, Mir I. Multi Robot Space Exploration: A Modified Frequency Whale Optimization Approach. AIAA Sci. Technol. Forum Expo. AIAA SciTech Forum 2022, 2022. https://doi.org/10.2514/6.2022-1416.

[133] Gul F, Mir I, Rahiman W, Islam TU. Novel Implementation of Multi-Robot Space Exploration Utilizing Coordinated Multi-Robot Exploration and

Frequency Modified Whale Optimization Algorithm. IEEE Access 2021. https://doi.org/10.1109/ACCESS.2021.3055852.

[134] Brodzicki A, Piekarski M, Jaworek-Korjakowska J. The whale optimization algorithm approach for deep neural networks. Sensors 2021. https://doi.org/10.3390/s21238003.

[135] Petrović M, Miljković Z, Jokić A. A novel methodology for optimal single mobile robot scheduling using whale optimization algorithm. Appl Soft Comput J 2019. https://doi.org/10.1016/j.asoc.2019.105520.

[136] Zan J, Ku P, Jin S. Research on robot path planning based on whale optimization algorithm. Proc 2021 5th Asian Conf Artif Intell Technol ACAIT 2021 2021:500–4. https://doi.org/10.1109/ACAIT53529.2021.9731150.

[137] Castillo O, Trujillo L, Melin P. Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots. Soft Comput., 2007. https://doi.org/10.1007/s00500-006-0068-4.

[138] Dao TK, Pan TS, Pan JS. A multi-objective optimal mobile robot path planning based on whale optimization algorithm. Int. Conf. Signal Process. Proceedings, ICSP, 2016. https://doi.org/10.1109/ICSP.2016.7877851.

[139] Gul F, Mir I, Abualigah L, Mir S, Altalhi M. Cooperative multi-function approach: A new strategy for autonomous ground robotics. Futur Gener Comput Syst 2022;134:361–73. https://doi.org/10.1016/J.FUTURE.2022.04.007.

[140] Dai Y, Yu J, Zhang C, Zhan B, Zheng X. A novel whale optimization algorithm of path planning strategy for mobile robots. Appl Intell 2022:1–15. https://doi.org/10.1007/S10489-022-04030-0/TABLES/4.

[141] Chhillar A, Choudhary A. Mobile robot path planning based upon updated whale optimization algorithm. Proc. Conflu. 2020 - 10th Int. Conf. Cloud Comput. Data Sci. Eng., 2020. https://doi.org/10.1109/Confluence47617.2020.9058323.

[142] Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer. Adv Eng Softw 2014. https://doi.org/10.1016/j.advengsoft.2013.12.007.

[143] Thanya T, Wilfred Franklin S. Grey Wolf Optimizer Based Deep Learning for Pancreatic Nodule Detection. Intell Autom Soft Comput 2023;36:97–112. https://doi.org/10.32604/IASC.2023.029675.

[144] Ahila A, Poongodi P, Bourouis S, Band SS, Mosavi A, Agrawal S, et al. Meta-Heuristic Algorithm-Tuned Neural Network for Breast Cancer Diagnosis Using Ultrasound Images. Front Oncol 2022;12. https://doi.org/10.3389/FONC.2022.834028.

[145] Nappu SW, Nappu JD, Zhang C. Path-relinking Grey Wolf Optimizer for Solving Operation Sequencing Problem. 2022 IEEE Int Conf Mechatronics Autom ICMA 2022 2022:1375–80. https://doi.org/10.1109/ICMA54519.2022.9856130.

[146] Yang C, Liu Z, Zhang W, Yue W. Cooperative Online Path Planning for UAVs Based on DMPC and Improved Grey Wolf Optimizer. Chinese Control Conf CCC 2022;2022-July:5008–13. https://doi.org/10.23919/CCC55666.2022.9901663.

[147] Wei L, Cai Z, Zhou K. Multi-objective Gray Wolf Optimization Algorithm for Multi-agent Pathfinding Problem. 2022 IEEE 5th Int Conf Electron Technol ICET 2022 2022:1241–9. https://doi.org/10.1109/ICET55676.2022.9824428.

[148] Gul F, Rahiman W, Alhady SSN, Ali A, Mir I, Jalil A. Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming. J Ambient Intell Humaniz Comput 2020. https://doi.org/10.1007/s12652-020-02514-w.

[149] Gul F, Mir I, Alarabiat D, Alabool HM, Abualigah L, Mir S. Implementation of bio-inspired hybrid algorithm with mutation operator for robotic path planning. J Parallel Distrib Comput 2022;169:171–84. https://doi.org/10.1016/J.JPDC.2022.06.014.

[150] Dong L, Yuan X, Yan B, Song Y, Xu Q, Yang X. An Improved Grey Wolf Optimization with Multi-Strategy Ensemble for Robot Path Planning. Sensors (Basel) 2022;22. https://doi.org/10.3390/S22186843.

[151] Kumar R, Singh L, Tiwari R. Comparison of Two Meta-Heuristic Algorithms for Path Planning in Robotics. 2020 Int. Conf. Contemp. Comput. Appl. IC3A 2020, 2020. https://doi.org/10.1109/IC3A48958.2020.233289.

[152] Euldji R, Batel N, Rebhi R, Kaid N, Tearnbucha C, Sudsutad W, et al. Optimal Backstepping-FOPID Controller Design for Wheeled Mobile Robot. J Eur Des Syst Autom 2022;55:97–107. https://doi.org/10.18280/JESA.550110.

[153] Mirjalili S, Mirjalili SM, Hatamlou A. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 2016. https://doi.org/10.1007/s00521-015-1870-7.

[154] Barrow JD, Davies PCW, Harper CL, Shortt D. Science and Ultimate Reality: Quantum Theory, Cosmology and Complexity. Am J Phys 2006.

[155] Khoury J, Ovrut BA, Seiberg N, Steinhardt PJ, Turok N. From big crunch to big bang. Phys Rev D - Part Fields, Gravit Cosmol 2002. https://doi.org/10.1103/PhysRevD.65.086007.

[156] Eardley DM. Death of white holes in the early universe. Phys Rev Lett 1974. https://doi.org/10.1103/PhysRevLett.33.442.

[157] Wald RM. The thermodynamics of black holes. Living Rev Relativ 2001. https://doi.org/10.12942/lrr-2001-6.

[158] Guth AH. Eternal inflation and its implications. J Phys A Math Theor 2007. https://doi.org/10.1088/1751-8113/40/25/S25.

[159] Zhu L, Lin J, Wang ZJ. A discrete oppositional multi-verse optimization algorithm for multi-skill resource constrained project scheduling problem. Appl

Soft Comput J 2019. https://doi.org/10.1016/j.asoc.2019.105805.

[160] Liu J, Wei J, Heidari AA, Kuang F, Zhang S, Gui W, et al. Chaotic simulated annealing multi-verse optimization enhanced kernel extreme learning machine for medical diagnosis. Comput Biol Med 2022. https://doi.org/10.1016/j.compbiomed.2022.105356.

[161] Ikram RMA, Dai H-L, Ewees AA, Shiri J, Kisi O, Zounemat-Kermani M. Application of improved version of multi verse optimizer algorithm for modeling solar radiation. Energy Reports 2022;8:12063–80. https://doi.org/10.1016/J.EGYR.2022.09.015.

[162] Iqbal MN, Bhatti AR, Butt AD, Sheikh YA, Paracha KN, Ashique RH. Solution of Economic Dispatch Problem Using Hybrid Multi-Verse Optimizer. Electr Power Syst Res 2022. https://doi.org/10.1016/j.epsr.2022.107912.

[163] Liang S, Zhang R, Bai Y. 3D Path Planning based on MMVO. Proc. 33rd Chinese Control Decis. Conf. CCDC 2021, 2021. https://doi.org/10.1109/CCDC52312.2021.9601774.

[164] Ghith ES, Sallam M, Khalil ISM, Serry M, Hammad SA. Real Time Implementation for Tuning PID Controller Based on Advanced Optimization Techniques for Micro Robotics System. Int J Eng Adv Technol 2021. https://doi.org/10.35940/ijeat.f3073.0810621.

[165] Ghith ES, Tolba FAA. LabVIEW Implementation of Tuning PID Controller

Using Advanced Control Optimization Techniques for Micro-robotics System. Int J Mech Eng Robot Res 2022;11:653–61. https://doi.org/10.18178/IJMERR.11.9.653-661.

[166] Jarray R, Al-Dhaifallah M, Rezk H, Bouallègue S. Path planning of quadrotors in a dynamic environment using amulticriteria multi-verse optimizer. Comput Mater Contin 2021. https://doi.org/10.32604/cmc.2021.018752.

[167] Jalali SMJ, Khosravi A, Kebria PM, Hedjam R, Nahavandi S. Autonomous robot navigation system using the evolutionary multi-verse optimizer algorithm. Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern., 2019. https://doi.org/10.1109/SMC.2019.8914399.

[168] Yang XS. A new metaheuristic Bat-inspired Algorithm. Stud. Comput. Intell., vol. 284, 2010. https://doi.org/10.1007/978-3-642-12538-6_6.

[169] Ajeil FH, Ibraheem IK, Humaidi AJ, Khan ZH. A novel path planning algorithm for mobile robot in dynamic environments using modified bat swarm optimization. J Eng 2021;2021:37–48. https://doi.org/10.1049/TJE2.12009.

[170] Perez J, Melin P, Castillo O, Valdez F, Gonzalez C, Martinez G. Trajectory optimization for an autonomous mobile robot using the bat algorithm. Adv Intell Syst Comput 2018;648:232–41. https://doi.org/10.1007/978-3-319-67137-6_25/FIGURES/9.

[171] Glover F. Tabu Search—Part I. ORSA J Comput 1989.

https://doi.org/10.1287/ijoc.1.3.190.

[172] Khaksar W, Hong TS, Khaksar M, Motlagh ORE. Sampling-based tabu search approach for online path planning. Adv Robot 2012. https://doi.org/10.1163/156855312X632166.

[173] Tuson A, Glover F, Laguna M. Tabu Search. J Oper Res Soc 1999. https://doi.org/10.2307/3010402.

[174] Xing L, Liu Y, Li H, Wu CC, Lin WC, Chen X. A novel tabu search algorithm for multi-AGV routing problem. Mathematics 2020. https://doi.org/10.3390/math8020279.

[175] Châari I, Koubâa A, Bennaceur H, Ammar A, Trigui S, Tounsi M, et al. On the adequacy of tabu search for global robot path planning problem in grid environments. Procedia Comput. Sci., 2014. https://doi.org/10.1016/j.procs.2014.05.466.

[176] Kumar S, Muni MK, Pandey KK, Chhotray A, Parhi DR. Path Planning and Control of Mobile Robots Using Modified Tabu Search Algorithm in Complex Environment. SSRN Electron J 2020. https://doi.org/10.2139/ssrn.3539922.

[177] Khaksar W, Hong TS, Sahari KSM, Khaksar M, Torresen J. Sampling-based online motion planning for mobile robots: utilization of Tabu search and adaptive neuro-fuzzy inference system. Neural Comput Appl 2019. https://doi.org/10.1007/s00521-017-3069-6.

[178] Panda MR, Priyadarshini R, Pradhan SK. Autonomous mobile robot path planning using hybridization of particle swarm optimization and Tabu search. 2016 IEEE Int. Conf. Comput. Intell. Comput. Res. ICCIC 2016, 2017. https://doi.org/10.1109/ICCIC.2016.7919636.

[179] Balan K, Luo C. Optimal Trajectory Planning for Multiple Waypoint Path Planning using Tabu Search. 2018 9th IEEE Annu. Ubiquitous Comput. Electron. Mob. Commun. Conf. UEMCON 2018, 2018. https://doi.org/10.1109/UEMCON.2018.8796810.

[180] Lopes HJM, Lima DA. Surveillance Task Optimized by Evolutionary Shared Tabu Inverted Ant Cellular Automata Model for Swarm Robotics Navigation Control. SSRN Electron J 2021. https://doi.org/10.2139/ssrn.3962774.

[181] Hliwa H, Daoud M, Abdulrahman N, Atieh B. Optimal Path Planning of Mobile Robot Using Hybrid Tabu Search- Firefly Algorithm. Int J Comput Sci Trends Technol 2018;6.

[182] Wong WK, Ming CI. A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications. 2019 7th Int. Conf. Smart Comput. Commun. ICSCC 2019, 2019. https://doi.org/10.1109/ICSCC.2019.8843624.

[183] Gholizadeh S., Barati H. A COMPRATIVE STUDY OF THREE METAHEURISTICS FOR OPTIMUM DESIGN OF TRUSSES 2012. https://www.sid.ir/paper/329375/en (accessed November 28, 2022).

[184] Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell Mag 2006;1:28–39. https://doi.org/10.1109/MCI.2006.329691.

[185] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986;13:533–49. https://doi.org/10.1016/0305-0548(86)90048-1.

[186] Karaboga D. An idea based on honey bee swarm for numerical optimization 2005.

[187] Chen D, Wang Z, Zhou G, Li S. Path Planning and Energy Efficiency of Heterogeneous Mobile Robots Using Cuckoo–Beetle Swarm Search Algorithms with Applications in UGV Obstacle Avoidance. Sustain 2022, Vol 14, Page 15137 2022;14:15137. https://doi.org/10.3390/SU142215137.

[188] Serrano-Pérez O, Villarreal-Cervantes MG, González-Robles JC, Rodríguez-Molina A. Meta-heuristic algorithms for the control tuning of omnidirectional mobile robots. Eng Optim 2020. https://doi.org/10.1080/0305215X.2019.1585834.

[189] Ajeil FH, Ibraheem IK, Sahib MA, Humaidi AJ. Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm. Appl Soft Comput J 2020. https://doi.org/10.1016/j.asoc.2020.106076.

[190] Petrović M, Jokić A, Miljković Z, Kulesza Z. Multi-Objective Scheduling of Single Mobile Robot Based on Grey Wolf Optimization Algorithm. SSRN

Electron J 2022. https://doi.org/10.2139/ssrn.4058009.

[191] Châari I, Koubâa A, Trigui S, Bennaceur H, Ammar A, Al-Shalfan K. SmartPATH: An efficient hybrid ACO-GA algorithm for solving the global path planning problem of mobile robots. Int J Adv Robot Syst 2014. https://doi.org/10.5772/58543.

[192] Xu S, Ho ESL, Shum HPH. A hybrid metaheuristic navigation algorithm for robot path rolling planning in an unknown environment. Mechatron Syst Control 2019. https://doi.org/10.2316/J.2019.201-3000.

[193] Farley A, Wang J, Marshall JA. How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion. Simul Model Pract Theory 2022;120:102629. https://doi.org/10.1016/J.SIMPAT.2022.102629.

[194] Severance C. Guido van Rossum: The early years of python. Computer (Long Beach Calif) 2015. https://doi.org/10.1109/MC.2015.45.

[195] Sundnes J. Introduction to Scientific Programming with Python. 2020. https://doi.org/10.1007/978-3-030-50356-7.

[196] Overland B. C++ Without Fear:A Beginner's Guide That Makes You Feel Smart. Pearson Education, Inc.; 2016.

[197] Moler C, Little J. A history of MATLAB. Proc ACM Program Lang 2020.

https://doi.org/10.1145/3386331.

[198] Zhang H, Yang T. A Simulation System for Testing Robotic Navigation Based on CoppeliaSim and ROS. 2021 6th Int. Conf. Control. Robot. Cybern. CRC 2021, 2021. https://doi.org/10.1109/CRC52766.2021.9620146.

[199] Montenegro G, Chacón R, Fabregas E, Garcia G, Schröder K, Marroquín A, et al. Modeling and Control of a Spherical Robot in the CoppeliaSim Simulator. Sensors 2022. https://doi.org/10.3390/s22166020.

[200] Bogaerts B, Sels S, Vanlanduit S, Penne R. Connecting the CoppeliaSim robotics simulator to virtual reality. SoftwareX 2020. https://doi.org/10.1016/j.softx.2020.100426.

[201] Chitta S, Sucan I, Cousins S. MoveIt! IEEE Robot Autom Mag 2012;19:18–9. https://doi.org/10.1109/MRA.2011.2181749.

[202] Malvido Fresnillo P, Vasudevan S, Mohammed WM, Martinez Lastra JL, Perez Garcia JA. Extending the motion planning framework—MoveIt with advanced manipulation functions for industrial applications. Robot Comput Integr Manuf 2023. https://doi.org/10.1016/j.rcim.2023.102559.

[203] Vyas DR, Markana A, Padhiyar N. Economic 6-DOF robotic manipulator hardware design for research and education. Mater Today Proc 2022. https://doi.org/10.1016/j.matpr.2022.03.140.

[204] Lavrenov RO, Magid EA, Matsuno F, Svinin MM, Sutakorn J. Development and implementation of spline-based path planning algorithm in ROS/gazebo environment. SPIIRAS Proc 2019. https://doi.org/10.15622/sp.18.1.57-84.

[205] Takaya K, Asai T, Kroumov V, Smarandache F. Simulation environment for mobile robots testing using ROS and Gazebo. 2016 20th Int. Conf. Syst. Theory, Control Comput. ICSTCC 2016 - Jt. Conf. SINTES 20, SACCS 16, SIMSIS 20 - Proc., 2016. https://doi.org/10.1109/ICSTCC.2016.7790647.

[206] Platt J, Ricks K. Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation. J Intell Robot Syst Theory Appl 2022. https://doi.org/10.1007/s10846-022-01766-2.

[207] Rivera ZB, De Simone MC, Guida D. Unmanned ground vehicle modelling in Gazebo/ROS-based environments. Machines 2019. https://doi.org/10.3390/machines7020042.

[208] Kashyap AK, Pandey A, Parhi DR, Singh Gour S. Trajectory tracking of single and multiple humanoid robots in cluttered environment. Mater Today Proc 2022. https://doi.org/10.1016/j.matpr.2021.12.558.

[209] Kashyap AK, Parhi D, Pandey A. Improved modified chaotic invasive weed optimization approach to solve multi-target assignment for humanoid robot. J Robot Control 2021. https://doi.org/10.18196/jrc.2377.

[210] Hohl L, Tellez R, Michel O, Ijspeert AJ. Aibo and Webots: Simulation, wireless

remote control and controller transfer. Rob Auton Syst 2006;54:472–85. https://doi.org/10.1016/J.ROBOT.2006.02.006.

[211] Echeverria G, Lassabe N, Degroote A, Lemaignan S. Modular open robots simulation engine: MORSE. Proc. - IEEE Int. Conf. Robot. Autom., 2011. https://doi.org/10.1109/ICRA.2011.5980252.

[212] Noori FM, Portugal D, Rocha RP, Couceiro MS. On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo? SSRR 2017 - 15th IEEE Int. Symp. Safety, Secur. Rescue Robot. Conf., 2017. https://doi.org/10.1109/SSRR.2017.8088134.

[213] Carpin S, Lewis M, Wang J, Balakirsky S, Scrapper C. USARSim: A robot simulator for research and education. Proc - IEEE Int Conf Robot Autom 2007:1400–5. https://doi.org/10.1109/ROBOT.2007.363180.

[214] Zhang HJ, Su ZB, Yang TT. Design of Team Formation Simulation System for Unmanned Ground Vehicles Based on USARSim and ROS. Zidonghua Xuebao/Acta Autom Sin 2021. https://doi.org/10.16383/j.aas.c200102.

[215] Balaguer B, Balakirsky S, Carpin S, Lewis M, Scrapper C. USARSim : a validated simulator for research in robotics and automation n.d.

[216] Janson L, Ichter B, Pavone M. Deterministic sampling-based motion planning: Optimality, complexity, and performance. Int J Rob Res 2018. https://doi.org/10.1177/0278364917714338.

[217] Kavraki LE, Švestka P, Latombe JC, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 1996. https://doi.org/10.1109/70.508439.

[218] Li Q, Xu Y, Bu S, Yang J. Smart Vehicle Path Planning Based on Modified PRM Algorithm. Sensors 2022;22:6581. https://doi.org/10.3390/s22176581.

[219] Pan QK, Fatih Tasgetiren M, Liang YC. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Comput Oper Res 2008;35:2807–39. https://doi.org/10.1016/J.COR.2006.12.030.

[220] Amiri P. Discrete Particle Swarm Optimization for Flexible Flow Line Scheduling. All Theses 2015.

[221] Ouaarab A, Ahiod B, Yang XS. Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 2014;24:1659–69. https://doi.org/10.1007/S00521-013-1402-2/FIGURES/5.

[222] Mantegna RN. Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Phys Rev E 1994;49:4677. https://doi.org/10.1103/PhysRevE.49.4677.

[223] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997. https://doi.org/10.1109/4235.585893.

[224] Igel C. No free lunch theorems: Limitations and perspectives of metaheuristics.

Nat. Comput. Ser., 2014. https://doi.org/10.1007/978-3-642-33206-7__1.

APPENDIX

Pseudocodes of metaheuristic algorithms

Table A.1: Pseudocode of genetic algorithmTable A.1

GENETIC ALGORITHM	
1	*Choose encode method*
2	*G ← 0*
3	*G_{max} ← Maximum generation*
4	*Initialize population*
5	**for** *(G < G_{max})* **do**
6	**for** *(i = 1 to maximum population)* **do**
7	*Evaluate fitness of individual i*
8	**end for**
9	*Selection*
10	*Crossover*
11	*Mutation*
12	*Move new individuals to population G + 1*
13	*G ← G + 1*
14	**end for**
15	**return** *best individual*

Table A.2: Pseudocode for ant colony optimization

ANT COLONY OPTIMIZATION	
1	*Initialize nodes and necessary parameters*
2	*Initialize pheromone level of each node*
3	*Define maximum iterations ITR*
4	**while** *(ITR > 0)* **do**
5	**for** *each ant k* **do**
6	*η_j ← heuristic function of the search space (fitness value)*
7	*Transition_probability[j] ← $p_{ij}^k(t)$*
8	*Select node with the highest $p_{ij}^k(t)$*
9	*Update pheromone level $\tau_{ij}(t+1)$*
10	**end for**
11	*ITR = ITR - 1*
12	**end while**
13	*Best solution ← solution with best η_j*
14	**return** *Best solution*

Table A.3: Pseudocode of particle swarm optimization

PARTICLE SWARM OPTIMIZATION	
1	*Initialize particle population size S*

```
2   GEN ← 0
3   Initialize GEN_max
4   for each particle i = 1,...,S do
5       Initialize particle's position x_i
6       Initialize particle's best known position to initial position: p_i ← x_i
7       Initialize particles velocity v_i
8       if fitness(p_i) > fitness(g) then
9           g ← p_i
10      end if
11  while GEN < GEN_max do
12      for each particle i = 1,...,S do
13          for each dimension d = 1,...,n do
14              r_1, r_2 ← random normal distribution in [0,1]
15              Update particle's velocity v_id
16              Update particle's position x_id
17          end for
18          If fitness(x_i) > fitness(p_i) then
19              p_i ← x_i
20              If fitness(p_i) > fitness(g) then
21                  g ← p_i
22              end if
23          end if
24      end for
25      GEN = GEN + 1
26  end while
27  Best solution ← solution with best η_j
28  return Best solution
```

Table A.4: Pseudocode of artificial bee colony algorithm

```
ARTIFICIAL BEE COLONY ALGORITHM
1   Initialize bee population size SN = number of employed bees = number of
    observer bees
2   Evaluate fitness of each bee f(sol)
3   Set best solution, solBest ← sol
4   ITR ← 0
5   Initialize ITR_max
6   while ITR < ITR_max do
7       for each employed bee i = 1,...,SN do
8           Select random solution and apply random neighbourhood
            structure
9           Determine the probability of each solution, p_i
10      end for
11      for each employed be do
```

```
12                      sol ← select solution with highest probability
13                      apply random neighbourhood structure
14                      If f(sol) < f(solBest) then
15                              solBest ← sol
16                      end if
17              end for
18              ITR = ITR + 1
19      end while
20      return Best solution, solBest
```

Table A.5: Pseudocode of firefly algorithm

```
FIREFLY ALGORITHM
1   Objective function f(x), x = (x_1, x_2 ..., x_d)
2   Initialize population size x_i (i=1,2,...,n)
3   Determine the intensity (I) of each firefly determined by f(x)
4   Initialize GEN_max
5   while (GEN < GEN_max) do
6       for (i = 1 to n) do
7           for (j = 1 to i) do
8               if (I_j > I_i) then
9                   Vary attractiveness with distance r via exp(−γr)
10                  move firefly i towards j
11                  Evaluate new solutions and update light intensity
12              end if
13          end for
14      end for
15      Rank fireflies and find the current best
16      GEN = GEN + 1
17  end while
18  return Best firefly
```

Table A.6: Pseudocode of cuckoo search algorithm

```
CUCKOO SEARCH ALGORITHM
1   Objective function f(x), x = (x_1, x_2 ..., x_d)
2   Initialize population of n host nests
3   ITR ← 0
4   Initialize maximum number of generations ITR_max
5   while (ITR < ITR_max) do
6       i ← Get a cuckoo randomly by levy flight
7       Evaluate fitness(i)
8       j ← choose a nest
```

```
        if fitness(i) > fitness(j) then
            replace j by the new solution
        end if
        Abandon a fraction(P_a) of worst nest and build new ones
        Keep the best nests
        Rank the nests and find the current best
        Pass the current best solutions to the next generation
        ITR = ITR + 1
    end while
    return Best nest
```

Table A.7: Pseudocode of whale optimization algorithm

```
    WHALE OPTIMIZATION ALGORITHM
    Initialize the whale population X_i(i = 1,2, … , n)
    Calculate the fitness of each whale
    X_best = the best search agent
    while (t < maximum number of iterations)
        for each search agent
            Update a, A, C, l and p
            if (p < 0.5) then
                if (|A| < 1) then
                    Update current agent via Encircling Prey
                else
                    Select a random agent (X_rand)
                    Update current agent via Search for Prey
            else
                Update search agent via Spiral Bubble-net
        end for
        Amend the position of whales that are outside the search space
        Calculate the fitness of each search agent
        Update X_best if there is a better solution
        t = t + 1
    end while
    return X_best
```

Table A.8: Pseudocode of grey wolf optimization

```
    GREY WOLF OPTIMIZATION
    Initialize the prey wolf population X_i(i = 1,2, … , n)
    Initialize a, A, and C
    Calculate the fitness of each search agent
    X_α = the best search agent
```

```
5    X_β = the second best search agent
6    X_δ = the third best search agent
7    while (t < maximum number of iterations)
8        for each search agent
9            Update the position of the current search agent
10       end for
11       Update a, A, and C
12       Calculate the fitness of all search agent
13       Update X_α, X_β, X_δ
14       t = t + 1
15   end while
16   return X_α
```

Table A.9: Pseudocode of multi-verse optimizer

MULTI-VERSE OPTIMIZER

```
1    Create random universes (U)
2    Initialize WEP, TDR, and Best Universe
3    SU ← Sorted universes
4    NI ← Normalize inflation rate (fitness) of the universes
5    while the end criterion is not satisfied do
6        Evaluate the fitness of all universes
7        for each universe indexed by i do
8            Update WEP and TDR
9            Black hole index ← i
10           for each object indexed by j do
11               r_1 ← random([0,1])
12               if r_1 < NI(U_i) then
13                   White hole index ← RouletteWheelSelection(-NI)
14                   U(Black hole index, j) ← SU(White hole index, j)
15               end if
16               r_2 ← random([0,1])
17               if r_2 < WEP then
18                   r_3 ← random([0,1])
19                   r_4 ← random([0,1])
20                   if r_3 < 0.5 then
21                       U(i,j) ← Best Universe(j) + TDR
                         × ((ub(j) – lb(j)) × r_4 + lb(j))
22                   else
23                       U(i,j) ←= Best Universe(j) - TDR
                         × ((ub(j) – lb(j)) × r_4 + lb(j))
24                   end if
25               end if
26           end for
```

27 **end for**
28 **end while**
29 **return** *Best Universe*

Table A.10: Pseudocode of bat algorithm

BAT ALGORITHM

1 *Objective function f*(x), x = $(x_1, \ldots\ldots, x_d)^{\mathrm{T}}$
2 *Initialize the bat population* x_i $(i = 1,2, \ldots, n)$ *and* v_i
3 *Define pulse frequency* f_{i} *at* x_{i}
4 *Initialize pulse rates* r_{i} *and the loudness* A_i
5 **while** *(t < Max number of iterations)* **do**
6 *Generate new solutions by adjusting frequency,*
7 *and updating velocities and locations/solutions [equations (2) to (4)]*
8 **if** *(rand >* r_{i}*)* **then**
9 *Select a solution among the best solutions*
10 *Generate a local solution around by flying randomly*
11 **end if**
12 *Generate a new solution by flying randomly*
13 **if** *(rand <* A_i *&* $f(x_{\mathrm{i}}) < f(x_*)$ *)* **then**
14 *Accept the new solutions*
15 *Increase* r_{i} *and reduce* A_i
16 **end if**
17 *Rank the bats and find the current best* x_*
18 **end while**
19 *Post-process results and visualization*

Table A.11: Pseudocode of tabu-search

TABU-SEARCH ALGORITHM

1 *Generate initial solution* (x_0)
2 *Initialize tabu list (TL ← [ ])*
3 *Current solution(x) ← initial solution* (x_0)
4 *Best solution(*x_{best}*) ← current solution(x)*
5 *Define maximum iteration (*ITR_{max}*)*
6 *Iteration (ITR) ← 0*
7 **while** *(*$ITR < ITR_{max}$*)* **do**
8 S_N *← Get neighbours of* x_{best}
9 **for** $S \in S_N$ **do**
10 **if** $S \notin TL$ *&&* $fitness(S) > fitness(x_{\mathrm{best}})$ **then**
11 $x_{\mathrm{best}} = S$
12 **end if**
13 **end for**

14 **add** *S to TL*
15 **end while**
16 **return** x_{best}