Journal of Systems Engineering and Electronics
Vol. 32, No. 6, December 2021, pp.1450 – 1462

Experimental study of path planning problem using EMCOA for a holonomic mobile robot

MOHSENI Alireza*, DUCHAINE Vincent, and WONG Tony

Department of System Engineering, École de Technologie Supérieure, Montreal QC H3C 1K3, Canada

Abstract: In this paper, a comparative study of the path planning problem using evolutionary algorithms, in comparison with classical methods such as the A* algorithm, is presented for a holonomic mobile robot. The configured navigation system, which consists of the integration of sensors sources, map formatting, global and local path planners, and the base controller, aims to enable the robot to follow the shortest smooth path delicately. Grid-based mapping is used for scoring paths efficiently, allowing the determination of collision-free trajectories from the initial to the target position. This work considers the evolutionary algorithms, the mutated cuckoo optimization algorithm (MCOA) and the genetic algorithm (GA), as a global planner to find the shortest safe path among others. A non-uniform motion coefficient is introduced for MCOA in order to increase the performance of this algorithm. A series of experiments are accomplished and analyzed to confirm the performance of the global planner implemented on a holonomic mobile robot. The results of the experiments show the capacity of the planner framework with respect to the path planning problem under various obstacle layouts.

Keywords: holonomic robot, path planning, evolutionary algorithm (EA).

DOI: 10.23919/JSEE.2021.000123

1. Introduction

Motion planning is a term that is used in addressing the problem of how to devise an algorithm that converts human rational orders into the low-level meaning of how to move. For several decades, many methods and algorithms have been proposed for the path planning problems in various environments with different landmarks and constraints. In spite of applying many modifications to classical approaches to allow a better performance in static environments, these methods are still criticized for their inept handling of motion planning in environments under complex conditions. Three common problems encountered with more classical methods are slowness, computational complexity, and trapping in local minimums. These problems have propelled designers to apply powerful mechanisms to the path planning problems for better performance. Among such mechanisms, evolutionary algorithms (EAs) have gained in popularity because of their strength in solving complex problems in most conditions. The most current classical approaches consist of developed branches of some standard methods such as the roadmap, cell decomposition, mathematical programming, and potential fields. These techniques can solve most motion planning problems, but the solutions are not limited to these methods, and combinations of them are usually used in developing better path planners. These techniques generally face problems related to [1]: (i) local minima: when there is a balance between repulsive and attractive fields; (ii) trap situation: this problem occurs when the robot passes between two objects or when the robot reaches a dead end, such as the inside of a U-shaped object or an area; (iii) oscillation: high-speed movement and sudden changes in direction in narrow passageways prompt unstable oscillations because the robot is subjected to repulsive forces from both sides of the passage. Whereas these problems may be solved using certain techniques, they do impose a degree of computational complexity and cost to the algorithm. The application of metaheuristic algorithms has recently gained acceptance in many fields because EAs do not require any information on the fitness function of the underlying problem in order to perform well. To alleviate the drawbacks of classical methods, EAs can either be employed directly or used with some probabilistic-based approaches, such as probabilistic roadmaps (PRM), to remedy the performance of these methods. In [2], a modified neural network was represented for a real-time path planning in non-stationary environments. The disadvantage of the proposed algorithm is that global knowledge of the workspace is always required. In 2011, a genetic algorithm (GA)-based controller was advanced for the motion planning problem, allowing the robot to identify sta-

Manuscript received November 10, 2020.
*Corresponding author.

tic and moving objects in the environment [3]. The proposed controller has the ability to reconstruct the current optimum path while moving toward the target. In 2012, a global path planning based on a probabilistic GA was proposed in partially unknown environments including moving objects [4]. In [5], a new approach was proposed for facing complex dynamic environments. This approach employed GAs to generate a global path based on prior information of the environment. In 2015, the work by [6] proposed a method that finds a sequence of actions using a fitness function that evaluates the actions executed in the current generation. In [7], an autonomous exploration strategy was developed by combining the simulation localization and mapping (SLAM) and Voronoi methods for mapping the environment and differential evolution (DE) algorithm for localization. Yu et al. [8] introduced a hybrid algorithm, a mix of DE algorithm and a group search optimizer (GSO), for path planning of UAVs in complex environments. In [9], a method for global path planning was presented using the DE algorithm and the Voronoi diagram for modeling the environment. In 2012, Jati et al. [10] introduced a hybrid mix of harmony search (HS) and bacterial foraging (BF) for multi-robot path planning. The proposed method inserts the chemo-tactic behavior of BF into the state of the HS for better stability. In 2013, the HS search was first modified using a quad-tree free space decomposition scheme, after which the algorithm was applied for a global motion planning in a grid-based environment [11]. In [12], a new path planner was planned for a moving target, which deployed the D* algorithm as an initial path planner. In [13], the authors introduced a strategy for the path planning of multi-robot multi-target in a dynamic environment. The proposed technique is a combination of two evolutionary methods. First, artificial bee colony was used to find initial paths, and then an evolutionary programming optimized those solutions to get a short collision-free path. In [14], a new path planning method developed based on deep Q-learning combined with a neural network which was trained by experience data when the robot intrudes into an unknown environment. This approach converges to an optimum strategy with less time and can find a path with fewer steps and greater average reward in an uncertain environment. In [15], an improved same adjacency crossover operator was proposed for the GAs to generate an optimal path by optimizing the energy consumption of the mobile robot in the static environment. In [16], authors suggested a method to create smooth paths by optimizing criteria for fitting and smoothness using DE under distinct modes of initialization of the population, selection pressure, exploration and exploitation during sampling, providing a data-driven planning framework for the comfortableness in riding. Li et al. [17] proposed a path planner using the firefly algorithm with self-adaptive population size for the mobile robot path planning problem. The introduced method adaptively changes the number of the firefly population to plan feasible solution in terms of path stability, convergence speed, and the algorithm running time. In [18], the ant colony optimization plus A* (ACO-A*) algorithm was introduced by mixing up the ACO and the A* search algorithm for path planning of autonomous underwater vehicles. Li et al. [19] introduced an adaptive quantum-behaved framework to improve the weakness of the particle swarm optimization (PSO) algorithm such as premature convergence for achieving the global optimal docking tasks. In many robotic applications the optimality is not only limited to finding an optimal path but also other criteria such as safety and smoothness might be considered. In [20], a DE algorithm was incorporated into the PSO algorithm to make a hybrid multi-objective method to solve the path planning problem. In [21], a multi-objective bare bones PSO was mixed up with DE to solve the path planning of mobile robot considering three indices of the path length, and the smoothness degree and the safety degree of a path. In [22], a framework was developed in which a multi-objective ACO algorithm is used to escalate the parameters of a fuzzy system. Then, this optimized fuzzy system is successfully applied to the problem of a wall-follower robot. In [23], an artificial potential field algorithm was used to discover all feasible paths in a discrete-grid environment, and then the multi-objective GA was employed to find an optimal solution among those initial paths in continuous space using five customized crossover and mutation operators. According to [24] the performance of the GA becomes much worse as the problem size increases or as there are moving or unmapped but static obstacles in the environment. However, other algorithms such as the cuckoo optimization algorithm (COA) perform better than the GA when the problem size increases [25]. Thus, the modified COA (MCOA) algorithm is considered to be evaluated whether it performs better in searching the optimal path in uncertain environments with unmapped obstacles.

The contribution of this work resides in the implementation and study of a modification to the MCOA [26] through extensive experimentation in the mobile robot application. This modification proposes a new immigration process for the MCOA in order to advance the algorithm performance. Also, the incompetency of the A* algorithm and the GA is discussed and analyzed in a robotic application.

The rest of this paper is structured in chronological sequence as follows: In Section 2, the proposed modifica-

tion and related path planning techniques are discussed for this work; Section 3 reviews the hardware set-up and the dynamics of youBot, as well as the navigation configuration used in this work; the results of the implementation of evolutionary algorithms on youBot for the path planning problem are presented in Section 4; Section 5 analyzes the inability of the employed algorithms to find an optimal path; Section 6 concludes and discusses the results.

2. The proposed approach

2.1 Path planning using EAs

Many different representations can be used for the path planning problem, one of which is the EA approach. This paper mainly focuses on MCOA as an optimization method to discover a collision-free path in both partially unknown and known environments.

2.1.1 Enhanced MCOA (EMCOA)

MCOA [26], a mutated and self-adaptive COA, is a generic population-based metaheuristic optimization algorithm, which employs a mutation operator to generate diverse population. This algorithm mimics the life style of the cuckoo birds. Cuckoos, as other brood parasitic birds, spawn eggs in the other birds' nests. The optimization procedure begins by initializing cuckoo mothers as population. Second, the cuckoo mothers begin making eggs in the host birds' roosts. An area with the highest rate of survival eggs is recognized as the best nest. The mature cuckoos build new societies and habitats and then the other cuckoos emigrate from their current dwellings to the best area. The process of egg laying, choosing a new area, and immigration to it constantly continue until most population get together around the best area. Briefly, the optimization process of the MCOA is as follows:

(i) Population initialization: generating the habitat matrix of size $N_{\text{pop}} \times N_{\text{var}}$. N_{pop} is the number of population and N_{var} is the number of the problem's variables.

(ii) A probability-based eggs assignment is proposed.

$$\left(n_{\text{eggs}_{\max}} - n_{\text{eggs}_{\min}}\right) \frac{\mathrm{e}^{-c_i}}{\sum_{i=1}^{n_c} \mathrm{e}^{-c_i}} + n_{\text{eggs}_{\min}} \tag{1}$$

where $n_{\text{eggs}_{\max}}$, $n_{\text{eggs}_{\min}}$ and n_c are the maximum and the minimum number of eggs, and the whole present numbers of cuckoos respectively. The term $\dfrac{\mathrm{e}^{-c_i}}{\sum_{i=1}^{n_c} \mathrm{e}^{-c_i}}$ is the Boltzmann distribution function and calculated for each cuckoo i. The allocation of eggs to cuckoos as a function of the Boltzmann distribution function increases the exploitation that ends up with finding a better solution instead of a random allocation of eggs to each cuckoo [27]. According to the nature of this bird, this egg laying process is limited to a maximum distance from the cuckoo mother nest called ELR and is defined as

$$\text{ELR} = \alpha \frac{n_{\text{eggs}_c}}{n_{\text{eggs}_t}} (v_h - v_l) \tag{2}$$

where n_{eggs_c} and n_{eggs_t} are the number of present cuckoo eggs and the total number of eggs, α is an integer which controls the maximum value of the egg laying radius, v_h and v_l are respectively the high and low bounds of the problem parameters [26].

(iii) Immigration process. The mature cuckoos immigrate to a better dwelling area on account of spawning eggs. The new position of the habitat after immigration is indicated [26] as

$$x_{\text{nh}} = x_{\text{ch}} + \{U(0,1) \min(|x_{\text{ch}} - v_h|, |x_{\text{ch}} - v_l|)\} \cdot (x_{\text{best}} - x_{\text{ch}}) \tag{3}$$

where x_{best} is an individual with the lowest value in the current iteration and $U(0,1)$ is a random number between zero and one and x_{ch} is the position of the current habitat. This modification ensures that the next habitat position is not outside the parameters' bound nor dose it perch on the boundary edge. Fig. 1 illustrates a geometric explanation of the motion coefficient boundaries. However, it should be considered that it is difficult to determine the boundary values for an actual problem with a high complexity level of the workspace such as the path planning problem for mobile robots. Therefore, these values are defined using the values in accordance with the geometry of the environment extracted from the map. It is a practical solution if the boundary values coordinate with the boundary areas of initial particles in the adaptive Monte Carlo localization (AMCL) layer. To decrease the probability of being stuck in local minima, a mutation operator is introduced in [26] as follows:

$$\dot{x}_c(t) = x_c(t) + \sigma(t) N(0,1) \tag{4}$$

where $\sigma(t)$ is the step size of the mutation, $x_c(t)$ is a mutated solution and N is the normal Gaussian distribution. For $\sigma(t)$ the simplest scenario is to set the step size value as a fixed value. The drawback of this approach is that a too small value or a too-large value diminishes the exploitation or the exploitation capability of the algorithm, respectively. Although it is possible to apply a self-adaptive scheme to the mutation step size, a fixed value is used here in order to maintain simplicity in the implementation of the algorithm. For example, the self-

adaptive mutation operator introduced in [26] suffers from heavy-computation processing when the geometry of the environment is complex. This work introduces a new motion coefficient (MC) as elaborated in (iv).

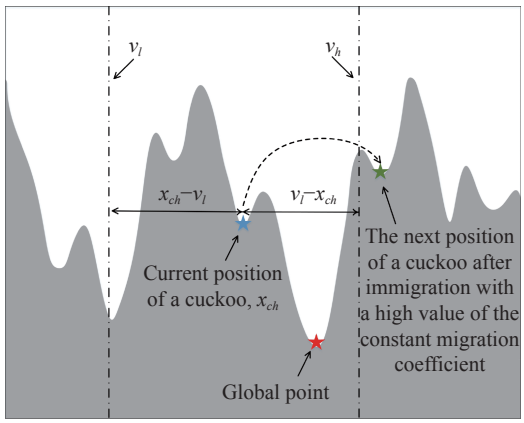


Fig. 1 A geometric explanation of MC boundaries

(iv) MC. MC is the main operator of the COA algorithm which plays a vital role in the performance of this algorithm such as accuracy, fast convergence, premature convergence avoidance, and the local minimum escaping. Equation (3) leads all current particles (cuckoos) of each group to randomly move towards the best particle (the position of the best current habitat). Although a random movement is a less complicated method and it gives an equal chance of selection to all individuals, it is a time-consuming process of research. In this case, there is no guarantee that the offered solutions are optimal and universal, especially when the geometry of the environment is complex and multimodal. Here the suggested immigration factor has a non-uniform scheme to ensure that the immigration process actively searches the problem workspace for finding the optimum solution. The H-spread measure is adopted to determine the pattern of distribution of particles within and outside the best current habitat. The advantage of the H-spread measure is that it does not make assumptions about the distribution of particles meaning that it does not depend on the standard deviation or mean of the data. The following modification helps maintain the fine-tuning capability of the exploration and the exploitation of the algorithm, ending up with conducting an efficient global and local search for the final solution. When the particles are located outside the region of the best current habitat, the immigration coefficient is greater than that when they are located inside the region of the best current habitat. This region is determined by the H-spread measure. When a particle is close to the best current solution, it needs a small MC value. Otherwise, a large value causes the particles to move toward a region far away from the current position of the optimum solution which reduces the performance of the algorithm. The offered modification is defined by

$$\mathrm{MC}=\begin{cases}\left|\left(1-\dfrac{U(0,1)}{b}\right)\right|^{b}, & x_{ch}>Q_1-1.5IQR \text{ or}\\ & x_{ch}<Q_3-1.5IQR\\ \left|\left(1-\dfrac{U(0,1)}{b}\right)\right|^{-b-1}, & \text{otherwise}\end{cases}\tag{5}$$

where $b\in(0.5,2]$ is a parameter defining the degree of non-uniformity, Q_1 and Q_2 denote the first and third quartiles of the particles of the best current habitat, and $IQR=Q_3-Q_1$, where IQR is the interquartile range, or a measure of statistical dispersion.

According to (5) and Fig. 2, when a particle (particle A or B) falls on a region where it is outside the neighbor group of the best habitat (Group 1) a large MC scheme is selected. Otherwise, a small value for MC is required because, for example, the particle (particle C) is getting close to the best current particle (the best current solution defined by the red star) within the region of the best habitat. A smaller value of the MC for particles while they are approaching the goal point increases exploitation capability of the algorithm.

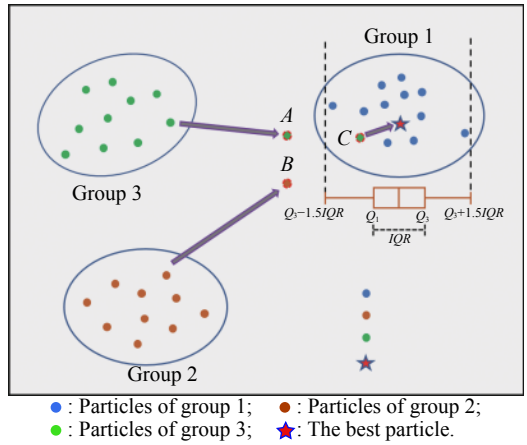


Fig. 2 Illustration of the mutated MC

For instance, choosing $b=1$ for (5) could lead to

$$\mathrm{MC}=\begin{cases}0.5, & x_{ch}>Q_1-1.5IQR \text{ or}\\ & x_{ch}<Q_1-1.5IQR.\\ 4, & \text{otherwise}\end{cases}\tag{6}$$

Equation (6) implies that a small value for the MC is assigned to particles close to the current global point.

2.1.2 GA

The GA is among the earliest forms of EAs that is based

on the genome conception. The main operators of the GA are selection, reproduction, mutation, and crossover. The preparation process of applying the GA for the path planning typically comes up with several phases: an apt chromosome depiction of the paths, separate mechanisms for both path guidance and static obstacle avoidance, and a suitable constrain definition. The main processes of the algorithm involved in the GA is as follows:

(i) Initialization. The population is typically randomly generated, which contains individuals, as a set of chromosomes.

(ii) Fitness function. Each individual of the population is then evaluated by how well the solution fits with the desired requirements.

(iii) Selection. To maintain proper diversity within the population and to avoid early convergence, parents are selected with high fitness and recombined to create offsprings for the following generation.

(iv) Crossover. This operator is similar to reproduction to create suitable children that inherit the best characteristic from their parents.

(v) Mutation. This operator is defined as adding randomness into the chromosome to increase genetic diversity. To avert the distortion of the highly fitted individuals, mutation usually is applied with a low probability.

The floating-point representation, which describes variables with a real-valued type, is used for describing the chromosome data type in order to maximize their sum. Generally, different chromosomal data types could be deployed to exploit the chromosome to achieve a better solution. The floating-point representation helps avoid premature convergence. The advantage of this representation is that an explicit encoding mechanism is not required. Since the chromosome representation directly affects the mutation or crossover operation performance, the chosen chromosomal data type could work worse or better for a specific problem such as a path planning problem. However, exploring the performance of other chromosome representations does not come within the purview of this work. In this paper, the GA did not implement within the robot operating system (ROS) framework as a node. Indeed, the GA interacts with the ROS instance using ZeroMQ messages. A message containing genome information is used by ROS to perform an evaluation. Upon completion of the evaluation, the fitness value as well as the genome ID will be returned to the GA. An ROS instance includes two nodes: adder-transporter and adder-worker nodes. The adder-transporter node manages the communication between the GA and the adder-worker node. The adder-worker node performs the actual summation of the genome and then returns the value to the adder-transporter node. Finally, fitness is sent back to the GA, along with the genome ID by the adder-transporter node.

2.2 Environment representation

There are two sources of information for a robot to use: the idiothetic and the allothetic sources. The former refers to self-proposition of the robot using the number of evolutions of the wheels, hinging on the cumulative error. The latter hints at the mounted sensors on the robot, such as a camera, a light detection and ranging (LiDAR), and so on. The problem of allothetic sources is that two different places can be noticed as the same. To tackle these deficiencies, a topological framework of the environment, which includes places and distance between them, is employed. A technique is needed to score possible trajectories, and so, in this work, the map grid is used. For each control cycle, a grid is generated around the robot location and then global path is partitioned into this area. Some certain cells with zero distance to the initial position and the goal point and the path point between them are marked. Then, all other cells are marked based on their Manhattan distance to those zero marked points.

The fitness function: The fitness function places importance on the algorithm's stability and performance such that an inadequate function may prompt the algorithm to either trap in local minima or oscillate around an optimum solution. Fitness functions are usually formed by the aggregation of weighted sub-functions including a path length sub-function and a collision avoidance term as a penalty. Equation (7) denotes this cost function f_{c} as follows:

$$f_{\mathrm{c}} = \sum_{i=1}^{n} pl_i - \sum_{j=0}^{n_{\mathrm{collision}}} \gamma \cdot \max(0, r_{\mathrm{o}} - d_j) \tag{7}$$

where pl_i is the distance between two sequence nodes n, d_j is the distance between the path and the edge of object, and r_o is the radius of the object. The value of γ, a collision constant for the penalty term, is considered such that no collision-free path is discovered when this value is too high. To keep balance between finding an optimal path and a collision-free path, the value of γ is one for all tests.

3. Experimentation

3.1 Experimental setup

The mobile robot such as the youBot from the company KUKA, which is a commercial product designed for research, incorporates the adroitness of a five-degree robotic arm into the adaptability of a mobile platform with a ca-

pacity to integrate several sensors to develop efficient algorithms for autonomous robotics purposes. The base platform hosts an on-board PC, an Intel Atom D510 Dual Core 1.66 GHz, a 2 GB Ram, a 32 GB SSD storage, and a 12 V DC Input. The youBot arm is equipped with a two-finger gripper. The arms can be mounted on the mobile platform and controlled by the on-board PC, or, they can be controlled via an Ethernet cable. Fig. 3 shows the youBot with an onboard laptop and a 2-D LiDAR mounted on the head of it. The base platform has four mecanum wheels, enabling the base platform to move in any direction. Table 1 indicates some detailed specifications of the robot base. Fig. 4 indicates the geometry of the base as A=74.87 mm, B=100 mm, C=471 mm, D=300.46 mm, E=28 mm [28].

Fig. 3 KUKA youBot equipped with a 2-D LiDAR and on-board laptop

Table 1 YouBot detailed base specifications

	Parameter	Value
Motor	Nominal voltage/V	24
	Nominal current/A	2.32
	Nominal torque/mN·m	82.7
	Moment of inertia/kg·mm^2	13.5
	Rated speed/rpm	5250
Gearbox	Reduction ratio	26
	Moment of inertia/kg·mm^2	0.14
Encoder	Counts per revolution	4000

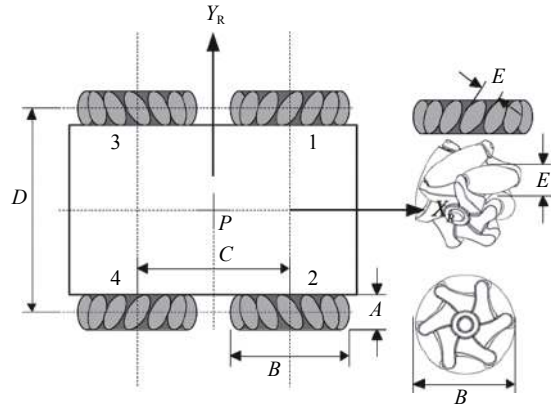


Fig. 4 Base geometry

3.2 Mecanum wheel

The youBot employs mecanum wheels, which allow the base platform to make rotational and transitional movements or a mix of both at the same time. It means that each wheel has 3 degrees of freedom (3-DOF) including the wheel rotation, the rolling rotation, and the rotational slip where it contacts with the ground.

Each mecanum wheel is composed of the six rollers attached to the circumference of the wheel center. All rollers are orientated at 45 from the rotation axis of the wheel.

3.3 Robot kinematics

The base's Jacobian matrix consists of four Jacobian matrices located on the axis of each wheel. The Jacobian matrix $\boldsymbol{J}_{\omega_i}$ for the wheel i is denoted [29] as

$$\boldsymbol{J}_{\omega_i}=\begin{bmatrix} R_i\sin\theta^R_{\omega_i} & r_i\sin(\theta^R_{\omega_i}+\eta_i) & d^R_{\omega_{iy}} \\ R_i\cos\theta^R_{\omega_i} & r_i\cos(\theta^R_{\omega_i}+\eta_i) & d^R_{\omega_{ix}} \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

where R_i is the perimeter of main wheel i, r_i is the roller's perimeter of the same wheel, and $\eta_1=\eta_2=-45°$ and $\eta_2=\eta_4=45°$. $d^R_{\omega_i}$ represents the distance between the robot's frame R and the wheel's frame i in Cartesian coordinate system. The movement of the mecanum wheels proceeds to the motion of the robot base. The final Jacobian matrix as a transform matrix for the velocity of the base is defined as

$$\boldsymbol{J}=\begin{bmatrix} \boldsymbol{J}_{\omega_1} & 0 & 0 & 0 \\ 0 & \boldsymbol{J}_{\omega_2} & 0 & 0 \\ 0 & 0 & \boldsymbol{J}_{\omega_3} & 0 \\ 0 & 0 & 0 & \boldsymbol{J}_{\omega_4} \end{bmatrix}. \tag{9}$$

3.4 Navigation configuration

Three fundamental components of the mobile robots' navigation system are the map builder, the motion and

local planners, and the platform controller. Fig. 5 shows the block diagram of navigation system's components implemented on the youBot. This navigation setup uses ROS environment. In the following, the components of this navigation system are described.

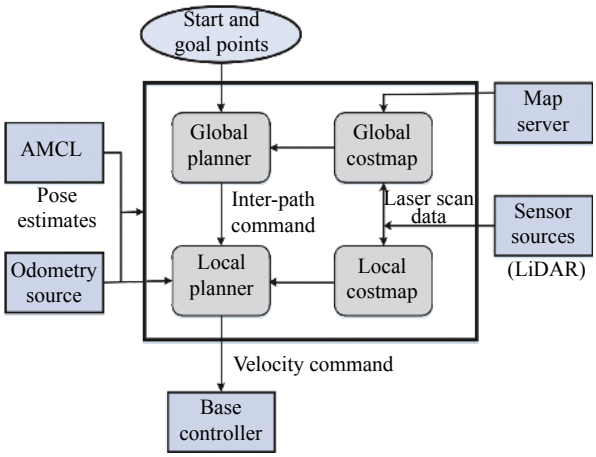


Fig. 5 Navigation setup for the youBot

(i) Sensor sources. Generally, the navigation system needs information obtained by sensors to avoid obstacles in the robot environment. A few sensor types can be used in this navigation system, two of which are laser light-based and point cloud-based sensors. In this work, a laser light sensor, Hokuyo URG-04LX-UG01, is used. This sensor detects objects by illuminating them with a laser light. As shown in Fig. 3, it has been mounted in the front of the youBot. Some main characteristics of this sensor are: detectable range d_r = is from 20 mm to 5 600 mm; measuring area $M_a = 240°$; scanning time $T_s = 100$ ms; angular resolution $r_a = 360^{\circ}/1\,024$ steps; noise N = 25 dB; power source $V = 5{\pm}5\%$ volts of direct current.

(ii) Odometry source. In robotics, motion sensors give data, known as odometry, to estimate changes in the position of the robot through time. The youBot has rotary encoders on its wheels which count 4 000 per revolution. These encoders measure the number of rotations by a wheel. In the youBot, the location of the robot is determined by transform frame (tf) and odometry source publishes transform and velocity information.

(iii) AMCL. AMCL is a probabilistic localization algorithm for a robot moving in the 2-D environment using a particle filter. The particle filter describes a distribution which estimates where the robot is. The algorithm distributes particles throughout the configuration space while the robot has no information on where it is. When the robot moves and receives information about the environment, it causes the particles to shift and predict the new state of the robot after it moves. When the robot senses an object, the particles are resampled using recursive Bayesian estimation to relate the actual captured data to the predicted state.

(iv) Costmap configuration. The costmaps stores information of obstacles in the robot's environment. A costmap is used for generating global planning throughout the robot's environment, and other costmap for localization and obstacle avoidance. The global costmap has several parameters such as global frame, which determines the frame in which the costmap should be executed, and the update frequency parameter, which defines the update loop frequency and the static map parameter indicates if the costmap needs to be initialized using the map served or not. The local costmap has the same parameters as the global costmap as well as others such as the rolling window. When this parameter is set to true, the costmap frame moves with the robot centered on it. The other parameters set the width, the height, and the resolution of the costmap.

(v) Base controller. The purpose of a control unit in a robotic system is to ensure that the system completes its tasks while it is self-reliant to perform in a complex environment. The control scheme is required for a reactively fast response to changes in real time as well as to maintain its stability and robustness. The base controller includes the robot driver and a proportional-integral-derivative (PID) controller, which rectifies the wheel velocities based on the difference error between the actual and desired velocity. At the upper level of abstraction, the defined target position and orientation of the robot with respect to some frame of reference original position is given to another ROS package, leading the robot to the goal position.

(vi) Map server. Map server is a ROS node that gives the stated information of a map via a ROS service used by the navigation system. This map server encodes the map image data into the occupancy values.

(vii) Global and local planners. The planner aims at creating a kinematic path for the robot to reach a goal location from a start position. This goal is achieved using a combination of two sub-planners: global and local planners.

A global planner is an algorithm that tries to find the most cost-effective path among all possible solutions. There are some famous methods such as the A* and Dijkstra's algorithms, which are the process of discovering a path between multiple nodes. It is known that these two algorithms are reasonably the optimal greedy algorithms for finding the shortest path in graphs search. A greedy algorithm is referred to any algorithm that adheres to the heuristic of finding the optimal solutions locally at each stage. In the A* algorithm, the search for an optimal solution is carried out among possible paths, and the ones that

appear to meet the optimality criteria are first considered. This method builds a tree of solutions starting from a particular staring point, known as a node, and then expands the paths until one of these paths reaches the predefined goal node. The best solution is the path that minimizes $f(n_l)=g(n_l)+h(n_l)$. n_l is the final node on the path, $g(n_l)$ is the total path cost from the start node to the final node, and $h(n_l)$ is a heuristic that estimates the smallest path cost from $g(n_l)$ to the goal position. Dijkstra's algorithm is also a method used to find the smallest trajectory between nodes in a graph. The algorithm comes in many variants: the original variant finds the paths between two nodes, while a more common alternative constructs a shortest-path tree between a node as a source and others.

In local planners, the goal is to localize the robot such that the robot safely avoids collision with obstacles. The dynamic window approach (DWA) and the trajectory rollout are two examples of the few number of techniques used to this end. The dynamic window approach considers the velocity space of the robot for handling control commands. This method involves the dynamics of the robot to reduce the complexity of the search space to those velocities that are safe with respect to the dynamic constraints and the limited accelerations of the robot. The second step of this method concerns the maximization of the objective function. This function includes three criteria: the distance to the nearest obstacle on the path, a measure which indicates the alignment of the robot with the goal direction, and the transitional and rotational velocities of the robot. The value of this function encodes the traversing costs through the grid cells. The controller's job is to employ this value to determine changes in velocity and then send an appropriate command to the robot. In each control cycle, a number of trajectories are generated, and then the collision-free trajectories are rated to select the best one.

4. Experimental results

One of the greatest challenges for a mobile robot is to avoid being trapping in a concave-shaped obstacle while facing some unmapped objects. This situation can be considered for the algorithm as trapping in a local minimum. In this section, experimental results using the youBot are shown. The average of the CPU running time of the EMCOA, the GA, and the A* algorithm is about 2.5 s, 2.4 s, and 6 s, respectively. They are measured at the beginning of each test when each algorithm tries to find the best global plan for the first time. The GA and the A* algorithm are implemented as base methods to be compared with the EMCOA algorithm. To evaluate the performance of the path planner, the configurations of two different environments are considered: known environments and partially unknown environments. In known environments all objects are fixed and mapped. In partially unknown environments, for a challenging assessment, some unmapped objects are added to the robot's workspace. To examine the effect of the proposed motion coefficient on the performance of the algorithm, a test is first conducted. As explained, (5) adaptively changes the MC value of particles corresponding to their distances from the best current particle location. This strategy mitigates the algorithm's susceptibility to the premature convergence and trapping in local minimums, and also the algorithm's inability to find an optimal path. The environment has two walls with an in-between split, followed by an angled corner where the target position is located. Fig. 6 makes a comparison of two path planner methods: the EMCOA algorithm (Path 1) and the original MCOA algorithm (Path 2) in a known environment. The known environment includes two walls with an in-between split followed by an angled corner.

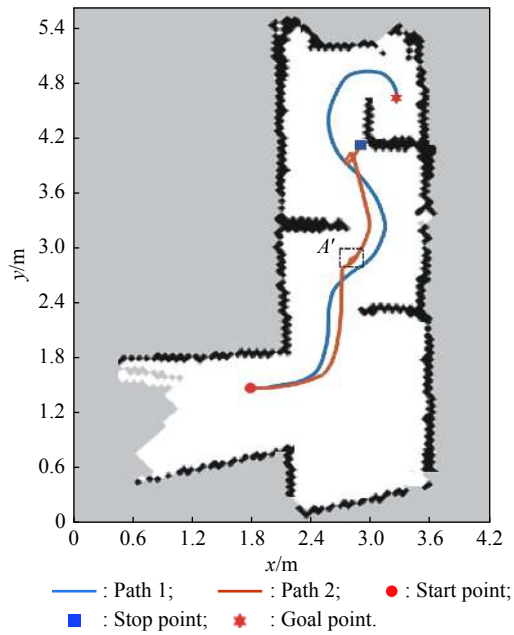


Fig. 6 Traversed path representation of the EMCOA algorithm compared with the MCOA algorithm in a known environment

The robot equipped with the EMCOA shows far better performance compared with the MCOA. The MCOA algorithm has a fixed value for the MC (MC = 0.8). The robot smoothly passes through two non-aligned splits with no oscillation (Path 1) and no premature convergence as the MC has an adaptive scheme. The MCOA algorithm could not generate a path allowing the robot to traverse the environment without getting stuck into the local minimum, since a low value for the MC was chosen (MC = 0.8); a low value of the MC provides less exploration

ability of the searching, causing the MCOA algorithm to converge a local solution. Before stopping at the end of Path 2 (indicated by the blue square in Fig. 6), the robot rotates around its center axis at point A' for few seconds to scan the environment and to regenerate a new path. In comparison with the EMCOA algorithm, this test demonstrates that the MCOA algorithm is incompetent to overcome the local minimum problem for the path planning. The EMCOA is used as a global planner for the rest of the paper. Table 2 lists the values for the EMCOA parameters.

Table 2 Values for parameters of the EMCOA algorithm

Parameter	Value
N_{pop}	14
v_l/m	0
$n_{eggs_{max}}$	6
v_h/m	5
$n_{eggs_{min}}$	3
a	5
b	1
σ	0.09

The inflation radius is set at 0.6 m to incur the cost of a safe path from obstacles. This implies that the robot considers all paths that remain 0.6 m or more away from obstacles are having equivalent obstacle costs. For this study, for the purpose of the optimality and smooth path planning, the value of the mutation rate for the EMCOA algorithm is kept low at 0.09 (σ in (4)) for all tests.

4.1 Known environments

For this situation, the algorithm was implemented on a robot that used a grid map built from laser data. In Fig. 7 and Fig. 8, the environment is assumed to be known with stationary obstacles. The global planner starts path planning based on information on the map while getting through the environment.

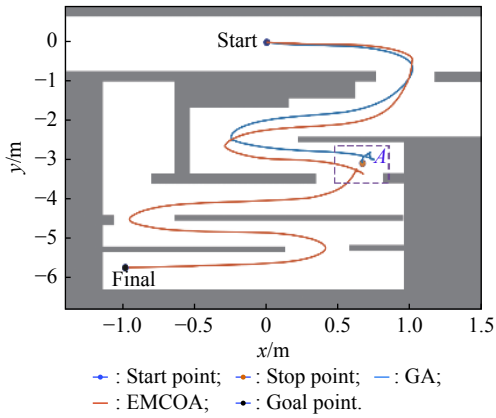


Fig. 7 Path-traversed representation of the EMCOA and GA in a maze environment

4.1.1 A narrow zigzag corridor

One of the most well-known limitations of some algorithms, such as the potential field methods, is the insufficiency of their motion stability while passing through a narrow passage. This instability usually occurs following a sudden disturbance, causing an oscillation reaction in the robot. This environment is employed here as a test plant to challenge the algorithms further. In this configuration, Fig. 7, the robot has to move from a relatively wide and long corridor to the other side through a narrow, twisty corridor. The start position $(0,0)$ and the goal point $(-5.76,-1.04)$ are outlined for the robot in the map. The width of the corridor is 70 cm on average which makes it a relatively narrow corridor as for the dimension of the robot according to Fig. 4. In area A, for the EMCOA, the robot changes its current orientation by moving backwards in the opposite direction to find a new feasible path to the goal, before resuming its navigation. In this area, such a maneuver occurs when the robot is guided by the genetic algorithm. However, the robot stops at coordinate $(0.66,-3.06)$, which constitutes a local minimum. When the GA is trapped into a local minimum, it means that the premature convergence occurs. Some techniques could be considered preventing the GA from reaching premature convergence such as increasing the population size, using a high crossover probability, and increasing the mutation rate. In the following test, the population size of the GA is significantly increased from fifty individuals (Fig. 7) to six hundred ones (Fig. 8). Also, the crossover probability value is increased to 0.95.

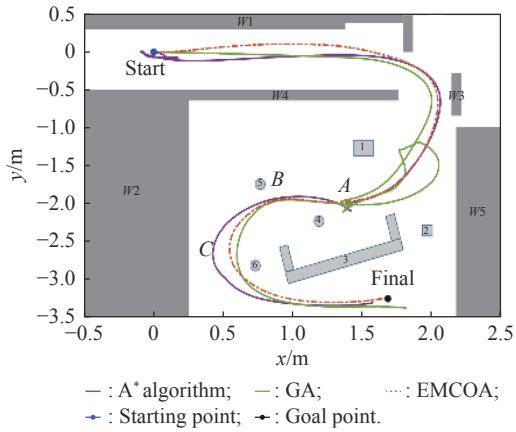


Fig. 8 Path-traversed representation of EMCOA, GA, and A* algorithms in a known environment

4.1.2 A concave shaped obstacle

As shown in Fig. 8, the environment includes a U-shaped obstacle at the bottom of the map, near the goal/final position. The environment also includes some objects with

different shapes and dimensions. The starting point $(0,0)$ and the goal point $(1.67,-3.24)$ are defined beforehand for the robot. The walls $W1$ to $W5$ and objects 1 to 6 are fixed and mapped. The worst-case scenario is when the robot reaches and gets inside the U-shaped obstacle but cannot get out of it. Otherwise, it needs to change its direction to find a sub-optimal path towards the final point. In the best scenario, the robot detects the obstacle and the path planner finds the shortest path, detouring the U-shaped object to the goal position. In the beginning, the robot has to first deal with the problem of passing through a relatively short but narrow corridor, and then a doorframe situation. In this condition, no going across closely spaced obstacles may happen or the robot turns away. Indeed, Fig. 8 represents an environment in which there is a mix of different situations, including a door frame, a narrow passage, a U-shaped obstacle, and a few different shaped objects. According to Fig. 8 and Table 3, the EMCOA algorithm could find the shortest collision-free path successfully while the robot is moving across the environment, with averaged run over thirty times. The A* algorithm could also generate a collision-free path, which is 12.3% or 1.45 m longer than the path planned by EMCOA. It takes also the robot 3.35 s more to follow the path created by the A* algorithm. On the first attempt, the GA algorithm fails to plan a collision-free path, allowing the robot to reach the goal. At point A, the algorithm starts re-planning to find a collision-free path by stepping back over the area it has passed, and then turning back to point A, and continuing on an onward path through objects to the final pose. Both GA and the A* algorithm fail to get to the final point. The GA exceeds it, whereas the A* algorithm cannot reach it. Although an increase in the population size and the crossover probability value helped the GA to escape from the local minimum, the path generated by this algorithm is not optimal at all; the performance of this algorithm still needs to be improved by deploying other factors such as tuning mutation rate, and deploying a preselection method to remove the same individuals from the population. In terms of video game path-finding, an absolute best path to the target is not needed; finding a fairly good path quickly is good enough. However, in realistic implementation of the A* algorithm such as robotics, there is a trade-off between the speed and accuracy. One of the widely used candidate functions for the heuristic, which is used in this study, is a linear function that represents a distance between two nodes using the Pythagorean theorem. Although this simple heuristic function helps the A* algorithm to perform fast, it is not a sufficiently precise function in order for the algorithm to find an accurately optimal path.

Table 3 Best-traversed time and path length associated with EMCOA, GA and A* algorithms (in Fig. 8)

Algorithm	Traversed time/s	Path length/m
EMCOA	21.46	6.68
GA	32.67	10.48
A*	24.81	8.13

4.2 Partially unknown environments

In most robotics applications, a navigation system needs to adopt a strategy that is sufficiently able to handle unexpected or unmapped obstacles. This test (Fig. 9) concentrates on the performance of the EMCOA and the analysis of its behavior in a complicated situation instead of demonstrating the behavior of the other used methods. However, the GA and the A* algorithm's performance is provided in terms of the path length and the traversed time compared with the EMCOA in Table 4 to show that the overall performance of the EMCOA is more effective than that of the GA and the A* algorithm. In this section, the global path planning was implemented in a partially unknown environment based on the initial information about the mapped objects in the environment and beginning and final goal positions. Since the details about some obstacles, such as their positions and dimensions, are unknown, it is not possible to draw an exact optimal path beforehand using only terrain information on the current existing map. When there is no dissimilarity between the prior map and the terrain, the robot follows the optimal path planned by the global planner. Otherwise, the global planner needs to cooperate with the local planner to modify the path to detour around the unmapped obstacles. Like Sub-section 4.1, the starting point and the goal point are defined beforehand for the robot. The walls $W1$ to $W5$ and Object 1 and Object 2 are fixed and mapped, but Objects 2, 4, 5, and 6 are unmapped. The robot does not have any prior information on the location and dimensions of unmapped objects.

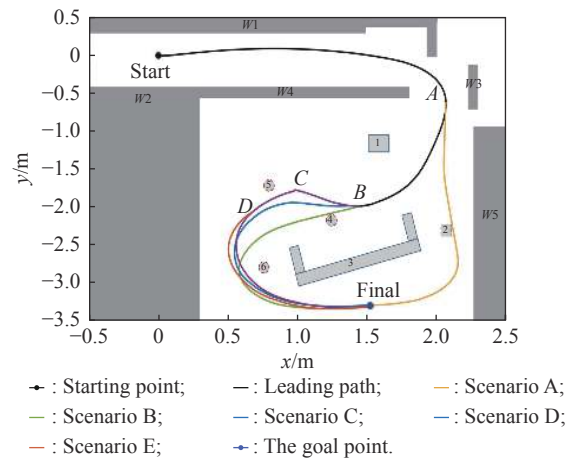


Fig. 9 Best-traversed paths using the EMCOA algorithm as a global path planner in a partially unknown environment

According to Fig. 9, the robot starts moving from the pose (0,0) and then goes through the corridor passage by following the black line planned by global planner without any oscillation. When it reaches point A, it could face the following different scenarios:

(i) Scenario A: If there are no unmapped objects, the planner leads the robot to the shortest path indicated by the yellow line.

(ii) Scenario B: If unmapped Object 2 exists, it is detected by the LiDAR sensor at point A, and so the planner generates a new path and the robot trails the black path and then the green line, up to the goal point. In this scenario, the robot is not trapped in the local minimum when faced with the U-shaped and unmapped objects.

(iii) Scenario C: Unmapped Object 2 and Object 4 are present. In this condition, Object 4 is detected at point B, and the planner generates a new path. The robot therefore pursues the blue path to stay away from the collision.

(iv) Scenario D: Unmapped Object 5 is added. There is almost no collision-free path between Object 3 and Object 4; therefore, the planner had to make a collision-free trajectory between Object 4 and Object 5. Point C is where that the robot reduces its speed with a small pause, changes its direction, passes through both objects successfully, and then reaches the final pose by following the purple path.

(v) Scenario E: In this situation, unmapped Object 6 is present along with the others. When the robot detects Object 6, it changes its direction at point D to avoid the collision with it, and then it follows its path indicated by the red line to the final point.

To evaluate the performance of the EMCOA, the GA and the A* algorithm are employed. Table 4 compares the length and time of the path traversed by the robot for each algorithm. The GA algorithm fails to give a solution as a path planner in a partially unknown environment when reaching Object 4. In contrast to the GA, the A* algorithm could generate a collision-free trajectory, guiding the robot to the final position with a longer traversed time and path length than those of the EMCOA.

Table 4 Traversed time and path length associated with three algorithms in a partially unknown environment

Algorithm	Traversed time/s	Path length/m
EMCOA	23.46	7.31
GA	>>45	N/A
A*	30.82	8.39

5. Discussion and analysis

It has been known for a long time that the A* algorithm has no limits on its performance, but the test illustrated by Fig. 8 shows that poor performance might happen in practice. The achievement of the A* algorithm relies significantly on the applied heuristic function. There are proper heuristics and improper heuristics for any given application that might be required to apply to the A* algorithm. A proper one would make it possible for this algorithm to run fast and find the optimal solution. An improper one could be so bad that it misleads the algorithm into finding sub-optimal solutions or even not finding any. An admissible heuristic guarantees that the algorithm discovers the optimal solution. When a heuristic is admissible, it does not misconstrue the cost of reaching the goal. It implies that an over-estimating heuristic considers the cost of an optimal solution higher than other sub-solutions; therefore, the optimal one will be overlooked in the selection of the best solution. The ROS navigation package uses Euclidean distance function for the heuristic in the A* algorithm. As presented results in Fig. 8, the A* algorithm needs a more accurate heuristic to discover the shortest path. However, in reality such a very accurate heuristic requires considerable computation, which is almost impossible to get.

As for the limitation of the GA, plenty of works have discussed the effect of parameters tuning on the performance of this algorithm [25,26]. In the GA, mutation operator is used to carry out the exploration; crossover operator is primarily used to lead the individuals to converge on the one of the found optimum solutions so far. Consequently, while the crossover tries to converge the algorithm into an optimal point in the workspace, the mutation aims at avoiding premature convergence and exploring the problem's landscape more. Although the mechanism of the mutation and crossover operators is extensively studied in constrained optimization problems, choosing the best values for these operators is very problem specific. One may consider other techniques such as Niching scheme or Crowding to maintain the diversity among species of the population.

The GA is a population-based algorithm that includes a set of chromosomes rather than a single solution, implying that compared with the EMCOA algorithm, it is more complex and difficult to implement. Thus, in some applications such as control and robotics, in comparison with the GA method, the EMCOA approach gives better trade-offs midst simplicity, precision, and computational cost. In view of complexity, EMCOA is less complex since it only deploys the immigration process as an opera tor.

6. Conclusions

In this paper, the efficiency of the EMCOA for the global path planning problem is studied. The main contribution consists of investigating a new scheme for the MC of the MCOA algorithm as compared to some well-known

classical algorithms. The EMCOA is employed for path planning in the grid environments and its performance is evaluated under different circumstances and situations, including those with mapped and unmapped objects.

In Section 4.1.1, Fig. 7 illustrates the simulation results for a typical local-minimum setting (zig-zag), during which the EMCOA has shown to be adequate to guide the robot to the target, leaving the local-minimum behind.

In Sub-section 4.1.2, the EMCOA shows its capacity as a global planner to tackle three environments set-ups comprising different shapes, such as U-shaped or cylinder objects. The experimental results indicate that the GA has difficulty generating a short collision-free path. The A* algorithm is successful in finding a collision-free path which is neither shorter nor more time efficient than the EMCOA.

In Sub-section 4.2, the capability of the path planners is evaluated in dealing with an uncertain environment consisting in unmapped objects and not having prior information on objects locations. Fig. 9 demonstrates that the EMCOA handles the path planning problem well when there are a few unmapped objects. The planner adaptively replans the shortest path step by step when an unmapped obstacle is detected by the LiDAR sensor, up to the final goal. The path generated by the EMCOA is shorter, and takes less time to traverse as compared to the GA and the A* algorithm.

It is shown that the EMCOA is fairly well-suited for finding an optimal solution for the path planning problem. Further work could focus on investigating some modifications in the local planner to improve its performance, especially for the localization in uncertain environments including noise in range sensor data.

References

[1] LAVALLE S M. Planning algorithms. Cambridge: Cambridge University Press, 2006.

[2] QU H, YANG X S, WILLMS A R, et al. Real-time robot path planning based on a modified pulse-coupled neural network model. IEEE Trans. On Neural Networks, 2009, 20(11): 1724–1739.

[3] YUN S C, PARASURAMAN S, GANAPATHY V. Dynamic path planning algorithm in mobile robot navigation. Proc. of the IEEE Symposium on Industrial Electronics and Applications, 2011: 364–369.

[4] YAZDANI H, FALLAH A, HOSEINI S M. A new approach for robot path planning with genetic algorithms. Journal of Basic and Applied Scientific Research, 2012, 4: 4122–4129.

[5] TAXIR M L, AZOUAUI O, HAZERCHI M, et al. Mobile robot path planning for complex dynamic environments. Proc. of the International Conference on Advanced Robotics, 2015: 200–206.

[6] DE CARVALJO SANTOS V, FABIANO MOTTA TOLEDO C, SANTOS OSORIO F. An exploratory path planning method based on genetic algorithm for autonomous mobile robots. Proc. of the 2015 IEEE Congress on Evolutionary Computation, 2015: 62–69.

[7] GARRIDO S, BLANCO D, MORENO L. SLAM and exploration using differential evolution and fast marching. InTechOpen, 2011.

[8] YU C Q, WANG Z R. UAV path planning using gso-de algorithm. Proc. of the 2013 IEEE International Conference of IEEE Region, 2013: 1–4.

[9] MO H W, MENG L L. Robot path planning based on differential evolution in static environment. International Journal of Digital Content Technology and its Applications, 2012, 6(20): 122–129.

[10] JATI A, SINGH G, RAKSHIT P, et al. A hybridisation of improved harmony search and bacterial foraging for multi-robot motion planning. Proc. of the IEEE Congress on Evolutionary Computation, 2012: 1–8.

[11] PANOV S, KOCESKI S. Harmony search-based algorithm for mobile robot global path planning. Proc. of the 2nd Mediterranean Conference on Embedded Computing, 2013: 168–171.

[12] DRAKE D, KOZIOL S, CHABOT E. Mobile robot path planning with a moving goal. IEEE Access, 2018, 6: 12800–12814.

[13] FARIDI A Q, SHARMA S, SHUKLA A, et al. Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment. Intelligent Service Robotics, 2018, 11(2): 171–186.

[14] JIANG L, HUANG H Y, DING Z H. Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge. IEEE/CAA Journal of Automatica Sinica, 2019, 7(4): 1179–1189.

[15] LAMINI C, BENHLIMA S, ELBEKRI A. Genetic algorithm-based approach for autonomous mobile robot path planning. Procedia Computer Science, 2018, 127: 180–189.

[16] PARQUE V, MIYASHITA T. Smooth curve fitting of mobile robot trajectories using differential evolution. IEEE Access, 2020, 8: 82855–82866.

[17] LI F L, FAN X J, HOU Z X. A firefly algorithm with self-adaptive population size for global path planning of mobile robot. IEEE Access, 2020, 8: 168951–168964.

[18] YU X, CHEN W N, GU T L, et al. ACO-A*: ant colony optimization plus A* for 3-D traveling in environments with dense obstacles. IEEE Trans on Evolutionary Computation, 2019, 23(4): 617–631.

[19] LI Z Y, LIU W D, GAO L E, et al. Path planning method for AUV docking based on adaptive quantum-behaved particle swarm optimization. IEEE Access, 2019: 78665–78674.

[20] ZHANG J H, ZHANG Y, ZHOU Y. Path planning of mobile robot based on hybrid multi-objective bare bones particle swarm optimization with differential evolution. IEEE Access, 2018, 6: 44542–44555.

[21] JUAN C F, LIN C H, AND BUI T B. Multi-objective rule-based cooperative continuous ant colony optimized fuzzy systems with a robot control application. IEEE Trans. on Cybernetics, 2018, 50(2): 650–663.

[22] JUANG C F, LIN C H, BUI T B. Multi-objective rule-based cooperative continuous ant colony optimized fuzzy systems with a robot control application. IEEE Trans. on Cybernetics, 2019, 115: 106–120.

[23] NAZARAHARI M, KHANMIRZA E, DOOSTIE S. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. Expert Systems with Applications, 2019, 115: 106–120.

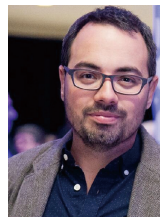
[24] YANG X S. Nature-inspired optimization algorithms. Waltham: Academic Press, 2020.

[25] MOHSENI A, DUCHAIN V, WONG T. A comparative study of the optimal control design using evolutionary algorithms: application on a close-loop system. Proc. of the Intelligent Systems Conference, 2017: 942–948.

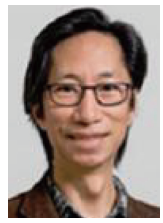
[26] MOHSENI A, WONG T, DUCHAIN V. MCOA: mutated and self-adaptive cuckoo optimization algorithm. Evolutionary Intelligence, 2016, 9(1/2): 21–36.
[27] RAJABIOUN R. Cuckoo optimization algorithm. Applied Soft Computing, 2011, 11(8): 5508–5518.
[28] Youbot. Youbot detailed specifications–detailed base geometry. 2014. http://www.youbot-store.com/wiki/index.php/File:MeasurementData.png
[29] DE GREEF F. Kuka youbot simulation. Amsterdam, Netherlands: University of Amsterdam, 2015.

Biographies

MOHSENI Alireza received his B.E. degree in electrical engineering from the Islamic Azad University (IAU), Tehran, Iran, in 2003, and his M.S. degree in mechatronics engineering from the IAU, Science and Research Branch, Tehran, in 2007. He was a researcher with the Multimedia University Cyberjaya, Malaysia, from 2008 to 2010. He is currently working toward his Ph.D. degree in automated manufacturing engineering focusing on robotics at École de Technologie Supérieure, Montreal, Canada. His research interests are industrial control systems, machine learning, computational intelligence, and robotic motion planning.
E-mail: ar.mohseni@yahoo.com



DUCHAINE Vincent received his B.E. and Ph.D. degrees in mechanical engineering from Université Laval, Quebec, Canada, in 2005 and 2010, respectively. He joined the Biomimetics Dexterous Manipulation Laboratory, Stanford University, Stanford, USA, as a post-doctoral fellow. He is one of the cofounders of Robotiq, Inc., a Canadian company that designs and manufactures flexible robotic grippers. He is currently a professor with the Department of Automated Manufacturing Engineering, École de Technologie Supérieure, Montreal, Canada. His current research interests include control and sensor design for improving intuitiveness and safety of physical human-robot interaction.
E-mail: vincent.duchaine@etsmtl.ca



WONG Tony holds his B.E. and M.E. degrees from École de Technologie supérieure in electrical engineering. He received his Ph.D. degree in computer engineering from École Polytechnique de Montréal. He joined the Systems Engineering Department of École de Technologie Supérieure in 1999. His research interests are multi-objective chance-constrained evolutionary algorithms, machine learning methods, and their application to service and manufacturing problems.
E-mail: tony.wong@etsmtl.ca