A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization — Source link

Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, Robin De Keyser

Institutions: Hanoi University of Science and Technology, University of Antwerp, Ghent University

Published on: 01 Oct 2017 - Soft Computing

Topics: Multi-swarm optimization, Any-angle path planning, Particle swarm optimization, Mobile robot and Multi-objective optimization

A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization

A Hierarchical Global Path Planning Approach for Mobile Robots based on Multi-Objective Particle Swarm Optimization

Thi Thoa Mac[a,b,*], Cosmin Copot[a,c], Duc Trung Tran[b], Robin De Keyser[a]

[a]*Department of Electrical energy, systems and automation, Ghent University, Technologiepark 914, 9052 Zwijnaarde, Belgium*
[b] *School of Mechanical Engineering, Hanoi University of Science and Technology, Dai Co Viet street 1, Hanoi, Vietnam*
[c]*Op3Mech, University of Antwerp, Salesianenlaan 90, B-2660 Antwerp, Belgium*

Abstract

In this paper, a novel hierarchical global path planning approach for mobile robots in a clutter environment is proposed. This approach has a three-level structure to obtain a feasible, safe and optimal path. In the first level, the triangular decomposition method is used to quickly establish a geometric free configuration space of the robot. In the second level, Dijkstra's algorithm is applied to find a collision-free path used as input reference for the next level. Lastly, a proposed particle swarm optimization called constrained multi-objective particle swarm optimization (CMOPSO) with an accelerated update methodology based on Pareto dominance principle is employed to generate the global optimal path with the focus on minimizing the path length and maximizing the path smoothness. The contribution of this work consists in providing a solution which combines classical algorithms with nature-inspired algorithm to reduce complexity and improve the quality of the robot path. Another advantage of such a hierarchical approach is computational efficiency and effective implementation to generate optimal paths. Simulation results in various types of environments are conducted in order to illustrate the superiority of the hierarchical approach.

Keywords: PSO, Multi-objective optimization, Pareto front, Constraints optimization, Mobile robot, Optimal path planning, Triangular decomposition, Dijkstra's algorithm.

1. Introduction

Recently, there has been an increased interest in developing intelligent mobile robot with advanced autonomous capabilities. The robot offers major advantages when used for reconnaissance, spatial and terrestrial explorations [1], harvesting [2, 3], cooperative formation [4, 5], coordinated manipulation of multi-robots [6], and decentralized multi-task distribution in multi-robot systems [7]. To accomplish the above mentioned missions, the prerequisite requirement is that the robot has to be able to handle various unexpected events that can disrupt the performance, thus global path planning becomes vital. The existing path planning methods are mainly grouped into two categories: *i*) classical algorithms and *ii*) heuristic-based algorithms [8], [9].

Prominent classical methods consist of cell decomposition method (CD) [10], potential field method (PFM) [11], roadmap method (RM) [12] and subgoal method (SG) [13]. Heuristic methods include neural network (NN) [14, 15], fuzzy logic (FL) [16] and nature-inspired methods from which the most famous ones are genetic algorithm (GA) [17] and particle swarm optimization (PSO) [18]. Each of the above approaches has its own limitations and so far, one individual method cannot perfectly solve the robot path planning problem. Thus, researchers have been patiently seeking for more powerful integrated methods for this problem. The aim of those methods is to figure out an optimal collision-free path from a starting position to a goal position under certain constraints. Since there are many types of robots with different characteristics, constraints and applications, it is nearly impossible to introduce an exact definition for the term "optimal path". However, it can be focused on several aspects, such as safety, smoothness, short distance approaches and energy with respect to constraints of changing direction and velocity. In other words, the robot path planning needs to be seriously considered as a constrained multi-objective optimization problem which contains more than one objective that needs to be achieved simultaneously [19]. Some of the suggested integrated methods are good for dealing with simple environments, however they work inefficiently and are time consuming for a clutter environments.

Consequently, this study proposes a novel hierarchical approach which combines triangular decomposition, Dijkstra's algorithm and CMOPSO. The interesting problem is the global path planning of a mobile robot evolving in such clutter environments. The aim of the approach is to generate optimal collision-free paths focused on minimizing the path length and maximizing the smoothness taking into account the robot's abilities. The proposed approach has a three level structure. In the first level, the triangular decomposition is used to divide the robot's working environment into obstacle configuration space and free configuration space. Next, in the second level, Dijkstra's algorithm is applied to find the collision-free path from the starting point to the goal point. Finally, CMOPSO with an accelerated update methodology based on Pareto dominance

*Corresponding author
Email address: thoa.macthi@ugent.be (Thi Thoa Mac)

principle is proposed to find the optimal path in terms of the path length and the path smoothness. Briefly, this work provides a novel solution which combines classical algorithms with nature-inspired algorithm to reduce complexity and improve the quality of the robot's path. Another advantage of such a hierarchical approach is computational efficiency and effective implementation to generate optimal paths.

The remainder of this paper is organized as follows: the related work is introduced in section 2 followed by section 3 where the single objective particle swarm optimization algorithm is presented in detail. Thereafter, the hierarchical path planing approach is described in section 4. The robot path planning formulation is given and explained in section 5. In section 6, the constrained multi-objective particle swarm optimization with a new update methodology based on Pareto dominance principle is described. Then, implementation of the proposed algorithm is described in section 7. Extensive simulations are carried out and performance evaluation of the proposed approach is discussed in section 8. Section 9 concludes with a summary of our contributions and a suggestion for future work.

2. Related work

As mentioned in previous section, the path planning methods are mainly divided into two categories : *i*) classical algorithms and *ii*) heuristic-based algorithms, as shown in Figure 1. Although such classical methods are often used in robot navigation tasks, there are still some challenges to deal with. For example, they do not take into account the robot's constraints and the obtained paths are not optimal.

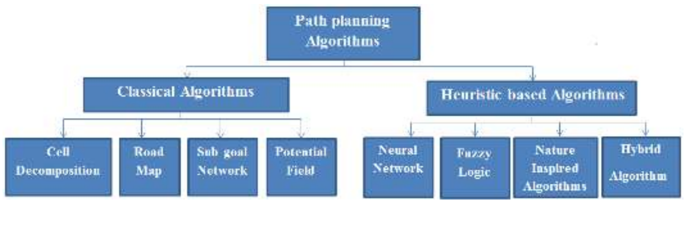


Figure 1: The classification of robot path planning algorithms.

Artificial neural network (ANN) have been applied to solve the path planning problem in modified forms. In [20], a self-adaptive auto-wave pulse-coupled neural network (SAPCNN) approach is applied for dealing with the shortest path problem. A novel contribution of the proposed SAPCNN is adjusting the propagation speed of the auto-wave adaptively according to the current state so that it spreads faster in finding the shortest paths. The experiments illustrate that SAPCNN outperforms classical algorithms. In [21], pulse coupled neural network (PCNN) is also employed for the all pairs shortest path problem (APSP). The authors proposed a novel parallel algorithm to solve APSP by a matrix multiplication method. This deterministic method guarantees the global solutions. A modified continued pulse coupled neural network (MCPCNN) model is proposed to solve two kinds of K shortest path (KSP) problems [22]. The method is able to find K shortest paths quickly by using the parallel pulse transmission characteristic of PCNN. However, ANN has several limitations: firstly, it requires an enough variety of training data and a possibly large learning cost with the increase in network structure. Secondly, ANN does not provide optimal paths.

Fuzzy logic is considered for expressing the subjective uncertainties in the human mind. A human has a notable capacity to perform navigation tasks without any exact measurements or computations. It is highly desirable to mimic this ability to develop autonomous robot navigation strategies [23]. In [24], J.H. Lilly proposed an approach using both negative fuzzy rules and traditional positive rules. In which, positive rules are stated to drive the robot to the goal in the absence of obstacles while a negative rules are activated in the presence of obstacles. As a result, fewer rules than using solely positive rules are applied on the obstacle avoidance controller. A fuzzy logic system with 48 fuzzy rules is presented in [25]. A combination of multiple sensors is equipped to detect the obstacles, the target and measure the current robot speed. It generates suitable paths toward the target in various scenarios without the "symmetric indecision" and the "dead cycle" problem. A positive aspect of the fuzzy approach is expressing navigation tasks in terms of linguistic variables which is simple and transparent. However, it has difficulty in selecting the most suitable rules and membership functions.

Nature-inspired algorithms are recognized to be properly suitable for solving optimization problems such as path planning because they can handle the complexity of multi-modality, nonlinearity and discontinuity which the robot often encounters in the autonomous navigation tasks. Therefore, those algorithms have recently received considerable interest from many researchers. Until now, a majority of research has been dedicated to looking for a feasible shortest path by formulating the robot path planning problem as a single-objective problem [26, 27], [28]. However, the optimized paths in real applications are not single-objective problems since many attributes such as path safety, path smoothness and minimum energy are also desirable. Unfortunately, very limited research into multi-objective robot path planning optimization has been conducted.

In [29], a multi-objective robot path planning has been developed to optimize the path length, the path safety and the path smoothness using Non-dominated Sorting Genetic Algorithm II (NSGA-II). To facilitate a discrete representation and for the navigation implementation, the grid-based approach is adapted in this study. However, the size of the robot is assumed to be smaller than unit cell size so that the dynamics of the robot is ignored and the obstacles are supposed to be the same size and shape. The disadvantage of this approach is that the computation increases dramatically as the number of obstacles grows. NSGA-II algorithm is also used for off-line path planning of unmanned aerial vehicles by applying B-spline curves. The objectives of minimizing the path length and maximizing the safety margin are also considered in [30].

PSO is a well-known nature-inspired algorithm because of its lower computational costs. The distribution of all publications and publication per year between 2010 and 2014 presented in [31] indicates that the number of total publications related to PSO is higher than the sum of other algorithms. This reveals that PSO is the most predominant swarm intelligent based optimization algorithm. The developments on PSO can be divided into the following five aspects: *i*) modifications of PSO such as quantum- behaved PSO, chaotic PSO, fuzzy PSO, topology; *ii*) hybridization of PSO with other heuristics methods such as GA, ant colony optimization (ACO), Tabu search (TS), artificial bee colony (ABC); *iii*) extensions of PSO to other optimization fields, consisting of multi-objective, constrained, discrete and binary optimization; *iv*) theoretical analysis of PSO with parameters selection and convergence analysis; *v*) parallel implementation of PSO, consisting of multi-core, cloud computing and GPU computing. PSO has been applied in many applications in numerous academic and industrial fields including electrical and electronic engineering, automation control systems, communication theory, operations research, mechanical engineering, medicine, biology, chemistry, fuel and energy [31].

Several papers compare PSO with GA showing that PSO can provide better performances in some cases [32, 33]. The existing studies have shown the capability of PSO to tackle the robot path planning in static and dynamic environments [34, 35]. In [36], the robot path planning with precise positions of the danger sources is solved by using multi-objective PSO. In [37], a new modified PSO named Stochastic PSO (S-PSO) is proposed which possesses a higher exploration ability than the original PSO so that a swarm with small size is employed to achieve the optimal path. With two objective functions with respect to trajectory's length and the obstacle avoidance, the algorithm is able to generate on-line, safe and smooth paths described by high order polynomials. In a similar approach, a hybrid of PSO and Gravitational Search Algorithm (GSA) was applied to find a short and safe path in dangerous environments [38]. A multi-objective path finding problem in a stochastic network is presented in [39], in which the parameters are divided into two types: deterministic variables and random variables. The authors proposed a chaos immune particle swarm optimization (CIPSO) which combines an artificial immune system (AIS), a chaos operator and PSO. Realistic computation shows that CIPSO has good performance in terms of route optimality and convergence time. Another interesting work which solves the unmanned combat air vehicle (UCAV) path planning problem by fitness-scaling adaptive chaotic PSO is presented in [40]. To improve the performance of the basic PSO algorithm, the authors propose three improvements: *i*) a new power-rank fitness-scaling method; *ii*) adaptive varied parameters to search an expansive area at the pro-phase stage and a restricted area at the anaphase stage; *iii*) apply chaos to improve the robustness of PSO. The performance evaluation function includes both the threat cost and the fuel cost in which the velocity of UCAV is supposed as a constant. The total cost for UCAV trajectory traveling is performed based on a weighted sum method.

The nature-inspired algorithms can deal with multi-objective robot path planning optimization problems however the drawback of those approaches is that they are time consuming and fail to create a feasible path in some scenarios. As updated trends, path planning approaches with hierarchical structures have been proposed in [41, 42, 43] to solve this type of problem. In those studies, the lower level principally focuses on obtaining a geometric collision-free path. To find such a path, graph search methods can be applied, such as A* [41, 42]. Then, the higher level is used to provide a series of subgoals to generate an optimal path. In [43], X. Yang et al. argue that due to the hierarchical structure design and the interpolative reasoning mechanism, the proposed path planning is very simple and concise. That brings several benefits, such as the reduction of computation time, re-usability of modules, and easy extensibility. In [44], a modified A^* algorithm, called Multi-Neuron Heuristic Search (MNHS) is implemented in a hierarchical manner where each generation of the algorithm provides a more detailed path with a higher reaching probability. The algorithm is able to give an optimal path in numerous situations with varying degrees of complexities where the standard A^* algorithm fails. Such a hierarchical approach is also very successful in structural engineering as presented in [45] using graph theory, Matroids and greedy algorithm for optimal cycle basis selection. In [46], size/topology optimization of trusses is proposed using GA, the force method and some concepts of graph theory. The approach is improved by using a suitable penalty function to reduce the number of numerical operations and to increase the speed of the optimization toward a global optimum.

In short, a hierarchical approach which combines several algorithms, for example classical algorithms with nature-inspired algorithms, reduces complexity of operations and can lead to impressive improvements in the area of robot application [47]. Another advantage of such a hierarchical methods is that they have computational efficiency and an effective implementation to generate an optimal path.

3. Single objective particle swarm optimization

PSO algorithm was firstly proposed by Eberhart-Kennedy [48] to solve the single objective optimization problem. PSO is inspired by social behavior of bird flocking or fish schooling. This algorithm is a population based stochastic optimization technique. In PSO, a swarm includes a set of particles and each particle is a potential solution of the optimization problem. Basically, PSO is initialized with a set of random solutions and then updated each generation based on an optimal scheme. Then, the global optimum is achieved by changing the collection of particles in a search space towards a promising area.

Considering the search space $\mathscr{D}$ that has dimension N ($\mathscr{D} \subset \mathscr{R}^N$), the position and velocity of the i^{th} particle in the swarm are $X_i = (X_{i1}, X_{i2}, ..., X_{iN}) \in \mathscr{D}$ and $V_i = (V_{i1}, V_{i2}, ..., V_{iN}) \in \mathscr{D}$. The particles will update their locations in the swarm towards the global optimum (or target position) based on two factors: 1) the personal best position (Pb) and 2) the global best position (Gb). The first term is the best position found by the i^{th} particle itself over iterations 1 ... t which is termed local leader and represented as $Pb_i(t)$= ($Pb_{i1}(t)$, $Pb_{i2}(t)$, ..., $Pb_{iN}(t)$). The second term is the best position of the whole particles in the swarm over iterations 1 ... t, which is termed global leader and represented as Gb. At the iteration $t+1$ of the search process, the velocity and the position will be updated according to following equations:

$$\begin{aligned} V_i(t+1) = wV_i(t) + c_1r_1(Pb_i(t) - X_i(t)) \\ + c_2r_2(Gb(t) - X_i(t)) \end{aligned} \tag{1}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{2}$$

where:
w is the inertia weight;
c_1 and c_2 are two nonnegative constants, referred to as cognitive and social factors, respectively;
r_1 and r_2 are uniform random numbers in [0, 1] that brings the stochastic state to the algorithm.
The pseudo code of this algorithm for minimizing a cost function J is provided in Algorithm 1.

Algorithm 1 PSO pseudo-code

```
Initialize population, parameters
While Termination criterion is unsatisfied
    For i=1 to Population Size
     Calculate particle velocity according to (1)
     Update particle position according to (2)
       If $J(X_i) < J(Pb_i)$
          $Pb_i = X_i$
          If $J(Pb_i) < J(Gb)$
             $Gb = Pb_i$
          End
       End
    End
End
```

The original PSO algorithm is designed to solve a single-objective optimization for a continuous solution space so that we must propose the particle representation, particle velocity and particle movement so that they work properly with multi-objective optimization for the robot path planning problem. The proposed method is described in sections 5 and 6.

4. The hierarchical robot path planning approach

In this section, the problem statement and important concepts are briefly introduced in the first subsection. Then, the framework of the proposed hierarchical path planning approach is presented in the next one. Finally, the triangular decomposition method and Dijkstra's algorithm also are shortly described.

4.1. Problem statement and definitions

First, the problem of path planning investigated in this study can be stated as follows: considering a cluttered environment, a start position and a goal position; path planning is to find a collision-free optimal path with a sequence of points that is safe and feasible for the mobile robot to follow. Without loss of generality, the obstacles are assumed to have convex polygonal shapes. From representation point of view, this assumption is not restrictive since any convex non-polygonal shape can be bounded by a convex polygonal region and a non-convex object can be divided into several convex objects. As the approach covers several notions related to the configuration space $\mathscr{C}$, some important definitions are introduced as follows.

Definition 4.1 (Working space): a working space $\mathscr{C}$ is a physical space that is a sub-set of $\mathscr{R}^2$ for planar $2\mathscr{D}$ or $\mathscr{R}^3$ for $3\mathscr{D}$ spaces.

Definition 4.2 (Obstacles configuration space): Obstacles are portions of $\mathscr{C}$ which are *occupied*, denoted by O_1, O_2, ..., O_n. The obstacles configuration space $\mathscr{C}\mathscr{O}$ is the mapping of the obstacles in the working space to the configuration space.

Definition 4.3 (Free configuration space): Free configuration space $\mathscr{C}_{free}$ is a set of configurations in which the robot is free from collision with obstacles:

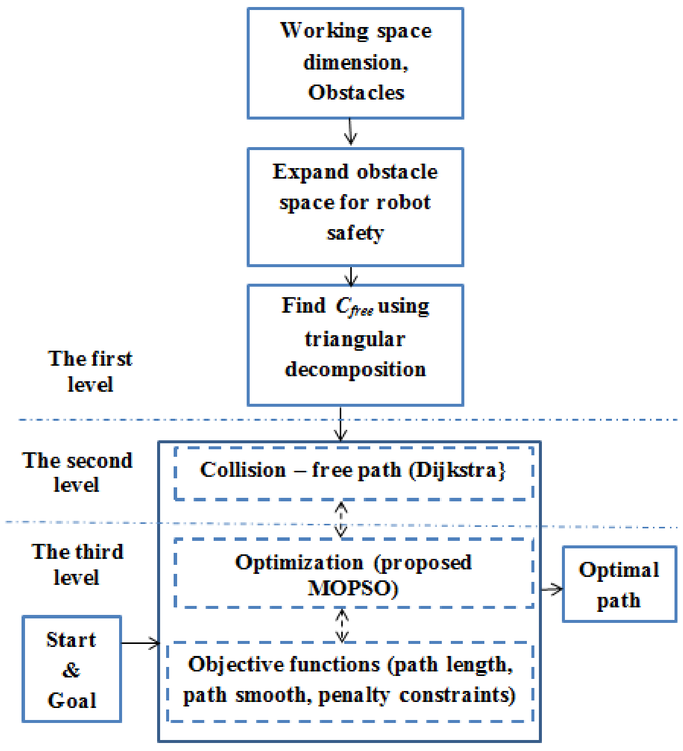
$$\mathscr{C}_{free} = \mathscr{C} \setminus \mathscr{C}\mathscr{O} \tag{3}$$

4.2. The framework of the proposed approach

In this subsection, the developed approach for the robot path planning problem is introduced. The key point is to develop multiple objective optimization based on PSO algorithm in a hierarchical manner. There are three different levels in the framework of the proposed approach presented in Figure 2. In the first level, the boundaries of every obstacles (in orange areas) are firstly expanded by an amount that is equal to the size of the robot plus a safety distance (in red areas) as shown in Figure 3. Therefore, the mobile robot can be considered as a point which freely moves out-side those boundaries. The triangular decomposition method is applied to quickly find the free space. In the second level, Dijkstra's algorithm is used to generate the collision-free path from the start location to the goal location (dashed red path). In the last level, a CMOPSO with an accelerated update methodology based on the Pareto dominance principle is applied to obtain the optimal robot path planning in terms of solution quality and actual execution time (solid blue path).

4.2.1. Triangular Decomposition

Cell decomposition is a partition of the free space into polygonal regions of the same type of geometry. The typical geometry are trapezoidal cells, triangular cells, polytonal cells, rectangular cells. In [10], the complexity and the quality of the path planning approach with respect to the chosen cell decomposition type are investigated. In that study, the results of decomposition types suggest that the triangular decomposition algorithm is the most advantageous choice from the point of view of combination of low number of cells, robustness to noise, small computation time and its large percentage of finding feasible paths. Thus, the triangular decomposition method is very promising and applied in this study.



Working space dimension, Obstacles
Expand obstacle space for robot safety
Find C_{free} using triangular decomposition
The first level
The second level
Collision – free path (Dijkstra)
The third level
Optimization (proposed MOPSO)
Optimal path
Start & Goal
Objective functions (path length, path smooth, penalty constraints)

Figure 2: The structure of the hierarchical robot path planning approach.

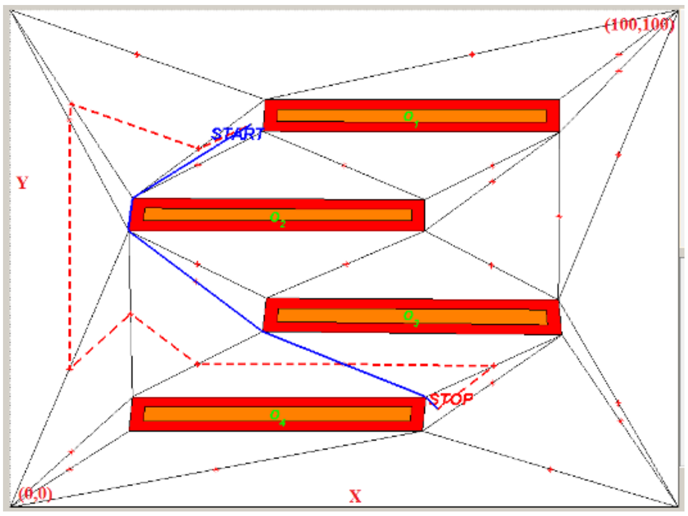


Figure 3: An example of the robot working space with four obstacles.

Basically, the triangular decomposition method has inputs $\mathscr{C}$ and $\mathscr{CO}$ while the outputs consist of a set of triangular cells $C = \{c_1, c_2, ..., c_m\}$ and the edges which correspond to adjacency among cells. The middle points of these adjacent edges are later used for Dijkstra's algorithm. Figure 3 is an example of the robot working space $\mathscr{C}$, 100 by 100 square meters which includes 4 obstacles. Points START and STOP respectively denote the starting and the goal positions. The white areas are free configuration space $\mathscr{C}_{free}$ and the red ones are $\mathscr{CO}$. Obviously, the configuration free space includes a set of triangular cells:

$$\mathscr{C}_{free} = \{c_1, c_2, ..., c_m\} \tag{4}$$

4.2.2. *Dijkstra's algorithm*

Dijkstra's algorithm is an efficient algorithm used to search the shortest path in a graph. In order to solve the robot path planning problem, a graph is constructed from the triangular decomposition where each node corresponds to a cell. The start cell is the one containing the start point and the destination one contains the goal point. Dijkstra's algorithm is employed to find an optimal sequence of the cells to be traveled. Then, the robot path is generated by linking the middle points of the line segments shared by successive cells from the sequence. For more details about that algorithm, please refer to [49].

4.2.3. *Constrained multi-objective particle swam optimization*

Evolutionary algorithms in general and PSO in particular may find an optimal path by themselves, however the disadvantage of those approaches is expensive computation. The aim of this step is to generate an optimal path based on the adjustment of the collision-free path points with the defined constraints by the proposed CMOPSO. More details on how the optimal path is generated will be given in sections 5 and 6.

5. Robot path planning formulation

In this study, the specific knowledge of the robot path planning problem is used to create proper initial particles, constraints and objective functions. To achieve this purpose, appropriate initial particles are created based on the results obtained from the combination between the triangular decomposition algorithm and Dijksra's algorithm. In addition, the modification of the original PSO is proposed to increase the speed of convergence, to evolve the robot constraints and to solve a multi-objective optimization problem. This section is divided into four subsections. The particles representation is firstly introduced (subsection 5.1). Then, the multi-objective path planning problem is defined (subsection 5.2), followed by the constraints problem (subsection 5.3). In the end, the problem formulation of the robot path planning is described (subsection 5.4).

5.1. *The particles representation*

Normally, the initial particles can be randomly created. This means a number of points are arbitrarily selected from the robot working space. However, this strategy has little chance to obtain a feasible path. When the environment becomes more complex, it will be more difficult to obtain a feasible one. Considering this problem, the combination of triangular decomposition and Dijkstra's algorithm is applied to ensure that all initial paths are free-collision ones.

As mentioned in section 4, the triangular decomposition method is employed to archive $\mathscr{C}_{free}$ of the robot, then Dijkstra's algorithm is implemented to find a collision-free path. One example shown in Figure 4, using Dijkstra's algorithm, the obtained collision-free path is $START \rightarrow P_1 \rightarrow P_2 \rightarrow ... \rightarrow P_8 \rightarrow STOP$ when the robot moves in the environment with three obstacles. Since the robot moves in the $2\mathscr{D}$ environment, each point is represented in x and y coordinates. For the path consisting of eight intermediate points, the corresponding particles representation is shown in Figure 4. The coordinates of the eight intermediate points are give by x_i and y_i ($i = 1...8$), which form particles denoted by X_i ($i = 1...16$).

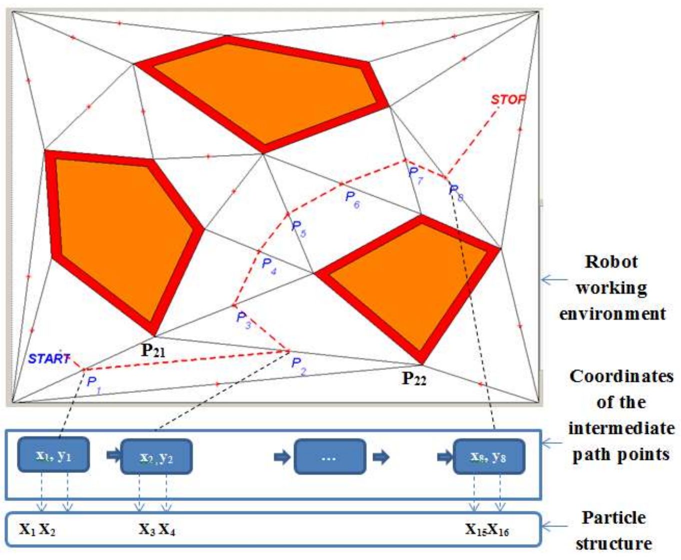


Figure 4: Solution representation of the robot path planning.

Obviously, each intermediate point P_i is the middle point of a line segment shared by adjacent cells. Assume, the relevant line segment has two vertexes termed P_{i1} and P_{i2} (for example P_2 is the middle point of the line with two vertexes termed P_{21} and P_{22} as shown in Figure 4). Thus, in general, for a path consisting of d intermediate points, the robot path planning problem is transformed into the following optimization problem.

Find a set of points $S = \{P_1, P_2, ..., P_d\}$ together with the start point and the goal point to create an optimal path where P_i (i = 1 ... d) satisfies the following coordinate constraints: $\min\{P_{i1}(x), P_{i2}(x)\} \leq P_i(x) \leq \max\{P_{i1}(x), P_{i2}(x)\}$ (i = 1 ... d) and $\min\{P_{i1}(y), P_{i2}(y)\} \leq P_i(y) \leq \max\{P_{i1}(y), P_{i2}(y)\}$ (i = 1 ... d). These constraints ensure that the robot is able to safely move in $\mathscr{C}_{free}$.

5.2. Multi-objective problem

The robot path planning is formulated as a multi-objective optimization problem with the aim of optimizing two objective functions while satisfying several inequality constraints. First, the objectives are formulated as follows:

Minimization of the path length:

Every path planning application must provide some degree of the shortest path. For the first performance criterion, the objective function is the total length of path, determined by:

$$\begin{aligned} J_1 = L(P) &= \sum_{i=0}^{d} L(P_i, P_{i+1}) \\ &= \sum_{i=0}^{d} \sqrt{(x_{i+1}-x_i)^2 + (y_{i+1}-y_i)^2} \end{aligned} \tag{5}$$

where $L(P_i,\, P_{i+1})$ presents the distance between two nodes P_i and P_{i+1}; x_i, y_i are the variable coordinates of node P_i (i = 0 ... d+1) in $\mathscr{C}_{free}$. In (5), P_0 ($START$) and P_{d+1} ($STOP$) represent the start point and the goal point with chosen coordinates.
In the particles form, equation (5) can be written as:

$$\begin{aligned} J_1 = L(X) = &\sqrt{(X_1-X_0)^2 + (X_2-Y_0)^2} \\ &+ \sum_{k=1}^{N} \sqrt{(X_{i+2}-X_i)^2 + (X_{i+3}-X_{i+1})^2} \\ &+ \sqrt{(X_{d+1}-X_{N-1})^2 + (Y_{d+1}-X_N)^2} \end{aligned} \tag{6}$$

where:
$N = 2d$;
X_0 and Y_0 are chosen coordinates of P_0;
X_i denotes particles.
X_{d+1} and Y_{d+1} are chosen coordinates of P_{d+1}.

Maximization of the path smoothness:

The path smoothness is evaluated by summing the robot's turning angle in the desired path. The smoothness of a trajectory is a very important attribute in robot path planning since the robot should not significantly change its direction. The path smoothness leads to lesser energy and time consumption. Therefore in this study, the smoothness is considered as a second objective function. Obviously, maximizing the path smoothness is equal to minimizing the total of the robot's turning angle. The cost function of the robot's turning angle is defined as follows:

$$J_2 = \Theta(P) = \sum_{i=1}^{d} |\theta_i| \tag{7}$$

where θ_i is the angle between two vectors $\overrightarrow{P_{i-1}P_i}$ and $\overrightarrow{P_iP_{i+1}}$:

$$\theta_i = atan2\left(\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right) - atan2\left(\frac{y_i-y_{i-1}}{x_i-x_{i-1}}\right) \tag{8}$$

In a similar way, equation (7) can be written in the particles form as follows:

$$\begin{aligned} J_2 = \Theta(X) = &\, atan2\left(\frac{X_4-X_2}{X_3-X_1}\right) - atan2\left(\frac{X_2-Y_0}{X_1-X_0}\right) \\ &+ \sum_{k=1}^{N-5} atan2\left(\frac{X_{i+5}-X_{i+3}}{X_{i+4}-X_{i+2}}\right) - atan2\left(\frac{X_{i+3}-X_{i+1}}{X_{i+2}-X_i}\right) \\ &+ atan2\left(\frac{Y_{d+1}-X_N}{X_{d+1}-X_{N-1}}\right) - atan2\left(\frac{X_N-X_{N-2}}{X_{N-1}-X_{N-3}}\right) \end{aligned} \tag{9}$$

5.3. Constraints problem

This section is devoted to define the constraints problem for the robot path planning. There are two considered factors:

the position constraints and the heading angle constraints of the robot.

The position constraints:

The generated points P_i with $i \in \{1, 2, ..., d\}$ are restricted by the coordinates of two points P_{i1} and P_{i2}. Thus, the lower bounds and upper bounds are:

$$min\{P_{i1}(x), P_{i2}(x)\} \leq P_i(x) \leq max\{P_{i1}(x), P_{i2}(x)\} \quad (10)$$

$$min\{P_{i1}(y), P_{i2}(y)\} \leq P_i(y) \leq max\{P_{i1}(y), P_{i2}(y)\} \quad (11)$$

Based on the definition of the particles presented in subsection 5.1, the generated points P_i can be expressed as:

$P_i(x) = X_{2i-1}$ and $P_i(y) = X_{2i}$.

Let's define:

$$\begin{aligned} X_{2i-1}^{LB} &= min\{P_{i1}(x), P_{i2}(x)\}; \quad X_{2i-1}^{UB} = max\{P_{i1}(x), P_{i2}(x)\} \\ X_{2i}^{LB} &= min\{P_{i1}(y), P_{i2}(y)\}; \quad X_{2i}^{UB} = max\{P_{i1}(y), P_{i2}(y)\} \end{aligned} \quad (12)$$

The general form of the position constraints problem is:

$$\begin{aligned} &LB_k \leq X_k \leq UB_k \\ &LB_k = (X_k^{LB}); \quad UB_k = (X_k^{UB}) \qquad \forall k \in \{1, 2, ..., N\} \end{aligned} \quad (13)$$

The heading angle constraints:

In the robot performance, it is difficult to rotate a large angle at one time because of the physical system limitations. Consider the changing direction has a range of $[-90^\circ \quad 90^\circ]$. This can be stated as an inequality constraint:

$$\psi_i(X) = |\theta_i(X)| - 90^\circ \leq 0 \quad \forall i \in \{1, 2, ..., d\} \quad (14)$$

Since the particles form of θ_i is mentioned in subsection 5.2, it is skipped in this subsection.

5.4. Problem formulation

Aggregating the proposed objectives and constraints, the global robot path planning can be mathematically formulated as a constrained multi-objective optimization problem as follows:

$$Minimize \quad [J_1(X), J_2(X)] \quad (15)$$

Subject to:

$$\psi_i(X) \leq 0 \qquad \forall i \in \{1, 2, ..., d\} \quad (16)$$

$$LB \leq X \leq UB \quad (17)$$

In this problem, there are no equality constraints.

6. Constrained multi-objective optimal particle swarm optimization algorithm

In order to solve the robot path planning problem described in section 5, a proposed CMOPSO algorithm is introduced in this section. First, the multi-objective optimization approach is presented. Then, the constrained method based on Pareto dominance principle is described, followed by the update of the archive, the global best and the particle position subsections.

6.1. Multi-objective optimization problem

For a multi-objective optimization problem (MOP), the most common approach is combining the optimization criteria into a single objective function by linear combinations of attribute values (called weighted sum method). This method is simple however it is problematic as the final solution depends on weighting factors.

Instead, the best trade-off solutions, called the Pareto optimal solutions or Pareto set, is the most powerful approach to solve the MOP. It is worth to note that multi-objective optimization provides a set of solutions (called Pareto-optimal) rather than one solution. This set includes the solutions that no solution is better than others with respect to all objective functions. In the following subsection the constrained dominance to deal with the multi-constraints in MOP is introduced. This relationship is used to update the feasible archive, the global best and the particle position.

6.2. Constrained Pareto dominance principle

For the constrained multi-objective robot path planning optimization problem, the dominance relationship which includes not only the objective function values but also its violation degree constraints is taken into account. It is reasonable to use its constraints-violated function to evaluate the violation degree. It means that each time the defined constraints is not satisfied, the violation degree increases according to following formulas:

$$vd = \frac{1}{d} \sum_{i=0}^{d-1} vd_j(X) \quad (18)$$

where:

$$vd_j(X) = \begin{cases} 0 & \text{if } \psi_i(X) \leq 0 \\ 1 & \text{if } \psi_i(X) > 0 \end{cases}$$

To handle efficiently constraints, an effective scheme is applied in the proposed algorithm. The constrained dominance principle is defined as:
(1) The solution with a smaller constrained violation degree is chosen to dominate the other.
(2) Two solutions have the same values of constrained violation degree, non-dominance principle is based on cost functions to choose the better solution.

6.3. Update the archives and the global best

For the robot path planning problem, a particle represents a potential path which connects the start point to the goal point and avoids all obstacles. Using the constrained Pareto dominance principle is a straightforward way to extend the basic PSO to handle multi-objective optimization problems. In this study, the update of the archives is based on this principle. At each iteration, the best solutions found by the swarm are compared on a peer to peer basis with elements in the archive which is set to empty at the beginning of the search. If the archive is empty, the current solution is directly stored. If the solution that the algorithm is wishing to enter, is dominated by any element in the archive then it is automatically discarded; otherwise, such a solution is stored in the archive. In addition, if any elements in the archive are dominated by the new element, then such old elements are removed from the archive [50].

The global best position (Gb) is the best solution obtained by all particles so far. In the proposed algorithm, the archive set is firstly found along the search process according to the above principle. Then the global best is chosen from the archive set by applying roulette wheel selection.

6.4. Update particle position

It is important to have a flexible search strategy based on the exploration ability and the exploitation ability. One approach is studied in [40], in which the parameters (w, c_1, c_2) change adaptively. In the search process of PSO, the search space will gradually decrease as the generation increases. In addition, a chaotic operator is applied to generate parameters (r_1, r_2). It is obvious that at the beginning of the search, the exploration ability is necessary to ensure that the algorithm can search in large space. Thereafter, the exploitation ability is preferred to guarantee that the algorithm can search promise areas carefully and converge to the optimal solution. In the conventional PSO, each particle updates its position based on both the current global best-*Gb* and the personal best-*Pb* (or local best) as mentioned in section 3.1. The purpose of using the local best is primarily to expand the diversity of the quality solutions, however, the diversity can be simply simulated by randomness at the beginning (for the exploration ability purpose) then reduces the randomness at each iteration (for the exploitation ability purpose). Therefore, in order to accelerate the convergence of the algorithm, it is possible to use the global best only [51].

The update strategy of the particle position is based on the accelerated particle updates method. The purpose is an increasing convergence speed toward global non-dominated solutions. Based on that statement, the velocity vectors are formulated as:

$$V_i(t+1) = V_i(t) + c_1 + c_2(Gb_i(t) - X_i(t)) \tag{19}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{20}$$

where:
c_1 is a random value in (UB-LB) that brings the stochastic state to the algorithm;
$c_2 \in [0.1\ 0.7]$; $Gb(t)$ is the global best in iteration t.
$V_i(t), V_i(t+1)$ are velocities of particles i in iteration $t, t+1$.
$X_i(t), X_i(t+1)$ are positions of particles i in iteration $t, t+1$.
To reduce the randomness as iterations are updated, the value of c_1 can be designed as:

$$c_1 = c_0 \gamma^t rand(UB - LB) \tag{21}$$

where $c_0 \in [0.1\ 0.5]$ is the initial value of the randomness parameter while t is the number of the iterations and $\gamma \in (0\ 1)$ is a control parameter.

6.5. Complexities of the algorithms

The computational complexity of the proposed algorithm is an important aspect that should be considered carefully. Measure of an algorithm complexity usually is the execution time, which can be estimated or predicted. Using quantity called steps is able to make the time measurement independent to a specific computer. The total number of steps is normally expressed as a function of the input size, called complexity function $T(n)$, where n is input size. Bounds of running time of Dijkstra's algorithm can be characterized as a function of the number of the edges, denoted $|E|$ and the number of vertices, denoted $|V|$. The algorithm running time is $O(|E|log(|V|))$ [52].

The number of computation required for a complete run of the PSO algorithm are the sum of the computations required to calculate the cost of a candidate solution (based on current position of the particles) and the computations required to update each particles position and velocity. Both of these are directly proportional to the number of iterations. In the robot path planning problem, the problem size is the number of intermediate path points. When apply the pure PSO algorithm, the problem size is a designed parameter, which depends on each scenario of working environment. To generate a feasible path in cluttered environment, this parameter is needed to set big enough (i.e. K=2* $the\ numbers\ of\ obstacle$ as each intermediate point has two coordinates x and y). The use of Pareto based pure PSO has led to program run times in $O(KT_{max}MS^2)$, where K is the number of intermediate points, T_{max} is the number iterations, M is the number of objectives, and S is the population size.

The proposed CMOPSO uses archives to store non-dominate solutions based on Pareto dominance principle. For the robot path planning problem, this requires time $O(NMSN_a)$ to test a candidate solution, while insertions and deletions can be done in constant time [53]; where N=2*d, d is the number of intermediate points as mentioned in section 5.1, N_a is the size of the archives. Obviously, the required time of the proposed method is much lower than pure PSO. Compared to pure graph algorithm, the computational time is also smaller. In addition, it provides an optimal path which cannot be done by Dijkstra's algorithm in the robot path planning application.

7. Implementation of the proposed algorithm

Based on the previous described content, the proposed constrained multi-objective particles swarm optimization algorithm for tackling the robot path planing is presented according to the following steps.

Step 1 (Model the working space) Establish $\mathscr{C}_{free}$ of the robot using the triangular decomposition method, then find a collision-free path $S = \{P_0, P_1, P_2, ..., P_d, P_{d+1}\}$ by using Dijkstra's algorithm; where P_0 and P_{d+1} denote the starting and the goal points of the robot.
Step 2 (Mathematical model) Build the mathematical model of the robot path planning optimization (particle representation) by the method proposed in Section 5.1. Consider the path has d intermediate nodes, each node has two coordinates x and y and the archived particles representation is $X = \{X_1, X_2, ..., X_N\}$ with $N = 2d$. Then, the objective functions and constraints are defined as in sections 5.2 and 5.3.
Step 3 (CMOPSO algorithm) Implement the proposed algorithm to find the mathematical model of robot path planning optimization problem.
Step 3.1 (Initialization) Set the necessary parameters (size of swarm-N, the size of the archives-N_a, the maximal sampling time T_{max}). The time (iteration) counter is set to 0 and the initial values are $c_0 \in [0.1\ 0.5]$, $c_2 \in [0.1\ 0.7]$. To increase the optimal convergence, an initial swarm X is randomly generated in [LB UB]. The pseudo code for this step is:

- For i = 1 to N
- Initialize $V_i = 0$
- Initialize X_i in [LB_i UB_i]

Step 3.2 (Evaluation) Calculate the objective functions and the constrained violated degree of each particles based on (6), (9), (18).
Step 3.3 (Update the archives) Search for non-dominated solutions and put in the archives.
Step 3.4 (Update the global best) The global best is selected from the archives.
Step 3.5 (Main loop) While ($t<T_{max}$),
(where T_{max}: predefined maximum iterations)

1. Update control parameter $c_1 = c_0 * \gamma^t$ where γ is a control parameter, chosen in (0 1).
2. Update velocity and position according to equations (19), (20).
3. Calculate the objective functions and the constrained violated degree of each particle based on (6), (9), (18).
4. Update the archives and the global best based on the principles introduced in section 6.3.
5. Update the iteration $t = t$+1

Step 4 (Display results) Display the optimal robot path.

8. Simulations and analysis

In this section, the proposed algorithm is validated through several test cases, assuming that a robot is performing respective mission in a 100x100 square meters working space. In each test, the obstacles are generated with different size, shape and the robot has different start and destination points. In the following simulations, the proposed algorithm used a set of parameters as: the swarm size $S = 60$, the size of archive $N_a = 20$; the maximum number of iterations $T_{max} = 60$, $c_0 = 0.2$; $c_2 = 0.7$; $\gamma = 0.97$.

(1) Test case 1

This test case includes two obstacles where their positions are not in the connection between the start and goal points. The positions of the start and the goal are (9.1, 83.6) and (80.7, 13.8), respectively. Ideally, the proposed algorithm should be able to find the straight line between those points.

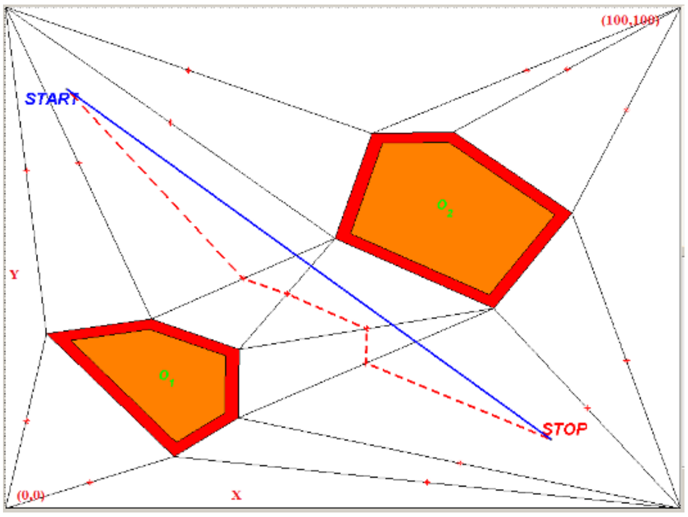


Figure 5: Obtained paths by Dijkstra's algorithm (dashed red line) and CMOPSO (solid blue line) in the test case 1.

For this test case, Figure 5 presents two obtained paths by Dijkstra's algorithm (dashed red line), and CMOPSO (solid blue line). Table 1 shows the coordinate values of those corresponding paths. For this problem, there are four intermediate points which form particles denoted by X_i (i = 1 ...8). In other words, the dimension of decision variable is 8. The position constraints which ensure that the robot is able to move freely in the working space are found as (according to subsection 5.1):

LB = [21.6, 37.8, 34.6, 31.7, 34.6, 31.7, 34.5, 18.1]

UB =[48.8, 53.8, 48.8, 53.8, 72.3, 39.9, 72.3, 39.9]

The heading angle constraints also are formulated according to equation (14). Dijkstra's algorithm is used to find the best direction from the start position to the goal position, then the proposed CMOPSO is applied to generate the optimal path focused on minimizing the path length and the path smoothness. It can be seen that the obtained path is the optimal solution for test case 1.

(2) Test case 2

Figure 6 displays a clutter space with 10 obstacles and shows two paths obtained by Dijkstra's algorithm (dashed red line) and CMOPSO (solid blue line) in the same approach as mentioned in test case 1. Dijkstra's algorithm finds the collision-free path from the start point to the target point via 8 intermediate points, thus the dimension of the decision variable is 16. The start and end points are (4.5, 45.9) and (93.4, 43.3), respectively. The results are summarized in the Table 2 for the position constraints and in the Table 3 for the trajectories generated by Dijkstra's algorithm and CMOPSO.

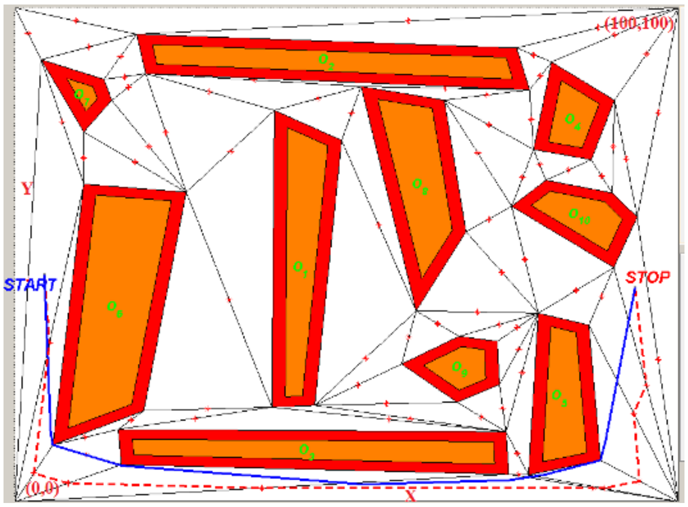


Figure 6: The multi−objective optimal paths obtained by Dijkstra's algorithm (dashed red line), CMOPSO (solid blue line).

(3) Test case3

The test case includes 20 obstacles with different sizes and shapes. The start position of the robot is (7.2, 92.6). Simulations in this scenario are performed for three different locations of the targets $STOP1$= (50.4, 22.6); $STOP2$ = (85.9, 72.2); $STOP3$ = (9.5, 4.7). The dimension of decision variable is 30, 24 and 14 respectively. The results are summarized in the Table 4 for the position constraints and in the Table 5 for the trajectories generated by Dijkstra's algorithm and CMOPSO. The simulations are shown in Figure 7. Figure 8 presents the approximated time consumption with different obstacles using CMOPSO. As can be seen from this figure, the proposed approach generates optimal solutions in the an extremely short computation time for environments with low, medium and high density (in terms of obstacles). The time consumption is less than 1 second for the environment with 20 obstacles.

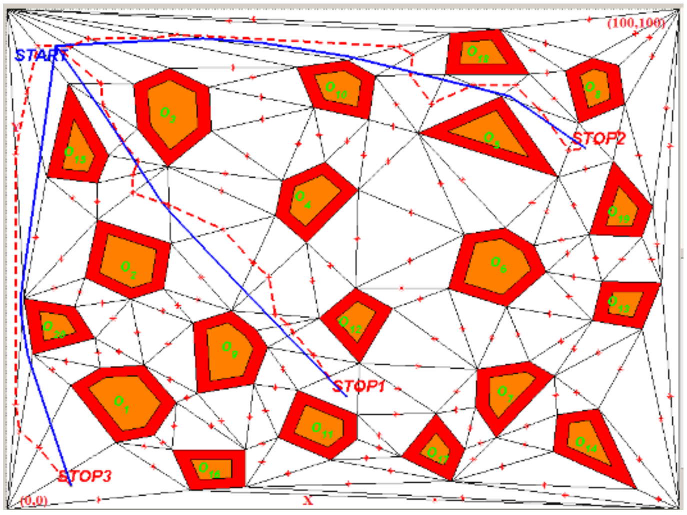


Figure 7: The multi-objective optimal paths obtained by Dijkstra's algorithm (dashed red line), CMOPSO (solid blue line) with the same start position and three different targets.

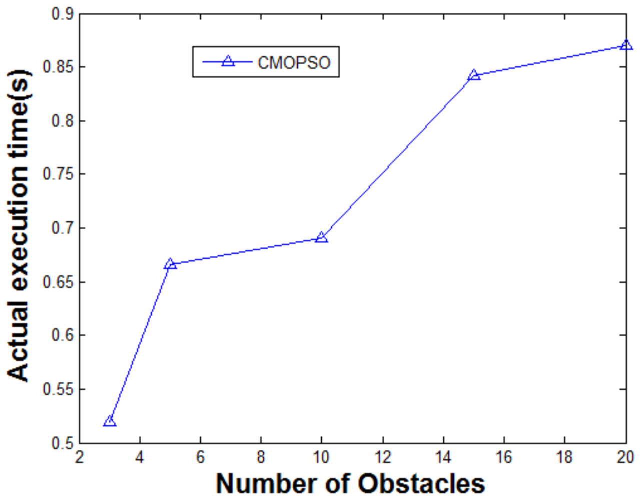


Figure 8: The actual execution time of CMOPSO with different number of obstacles.

9. Conclusions

In this study, a novel hierarchical approach for robot path planning in the presence of clutter obstacles is proposed. In this approach, a combination of the triangular decomposition method, Dijkstra's algorithm and a proposed constrained multi-objective particle swarm optimization (CMOPSO) with an accelerated update methodology based on the constrained Pareto dominance principle is proposed to get optimal path planning results. The resulting algorithm has a three level structure. In the first level, the triangular decomposition is used to swiftly find the obstacle configuration space and free configuration space

Table 1: The obtained paths of the test case 1

Paths	The coordinate of intermediate points (x,y)
Dijkstra's	(35.2, 45.8)→(41.6, 42.8)→(53.4, 35.8)→(53.4, 28.9)
CMOPSO	(43.5, 50.0)→(48.0, 45.6)→(61.9, 32.1)→(64.0, 30.1)

Table 2: The robot position constraints of the test case 2

The position constraints	
LB	[0, 0, 0, 0, 0, 0, 0, 0, 74.4, 0 , 77.3, 0, 88.2, 0, 86.5, 0, 90.2, 0]
UB	[10.3, 64.3, 5.7, 11.5, 16.1, 7.5, 74.4, 5.6, 100, 5.6, 100, 5.4, 100, 8.4, 100, 35. 8, 100, 74.4]

Table 3: The obtained paths of the test case 2

Paths	The coordinate of intermediate points (x,y)
Dijkstra's	(5.2, 32.1)→(2.9, 5.7)→(8.0, 3.8)→(37.2, 2.8)→(87.2, 2.9)→(88,7, 2.7)→(94.1, 4.2)→(93.2, 17.9)→(95.1, 23.7)
CMOPSO	(5.4, 13.8)→(5.7, 11.5)→(16.0, 7.3)→(52.5, 3.5)→(74.4, 4.3)→(79.7, 5.4)→(88.2, 8.4)→(89.0, 12.5)→()

Table 4: The robot position constraints of the test case 3

Targets	Types	The position constraints
$STOP1$	LB	[9.1, 84.9, 9.1, 76.1, 15.1, 70.7, 15.1, 68.5, 13.5, 65.2, 13.5, 57.9, 23.9, 53.1, 24.8, 53.1, 33.3, 39.7, 33.3, 39.7, 33.3, 38.2, 37.2, 35.8, 38.7, 32.3, 38.7, 29.1, 43.2, 23.8]
	UB	[18.8, 85.2, 19.4, 85.1, 19.4 76.1, 23.9, 70.7, 23.9, 68.5, 23.9, 68.5, 24.4, 68.5, 39.9, 58.7, 39.8, 58.6, 44.5, 53.3, 46.3, 39.7, 46.4, 38.2, 46.4, 38.2, 52.4, 32.3, 52.4, 29.2]
$STOP2$	LB	[0, 84.9, 0, 88.2, 0, 88.3, 0, 89.8, 50.6, 89.7, 54.7, 86.6, 54.7, 86.6, 61.1, 75.2, 65.1, 82.5, 73.5, 82.5, 73.5, 81.9, 66.4]
	UB	[18.7, 100, 24.2, 100, 43.1, 100, 50.7, 100, 65.9, 95.2, 65.8, 95.2, 65.2, 86.9, 65.1, 86.9, 73.5, 86.9, 77.6, 86.9, 84.9, 82.5, 84.9, 77.1]
$STOP3$	LB	[0, 85.2, 0, 66.4, 0, 42.1, 0, 0, 0, 0, 0, 0, 0, 0]
	UB	[9.1, 100, 5.9, 100, 2.54, 100, 2.5, 42.2, 3.8, 30.9, 9.4, 24.8, 16.2, 13.3]

Table 5: The obtained paths of the test case 3

Targets	Paths	The coordinate of intermediate points (x,y)
$STOP1$	Dijkstra's	(13.9, 85.1)→(14.3, 85.1)→(14.3, 80.6)→(17.3, 73.4)→(19.5, 69.6)→(18.7, 66.8) →(18.7, 63.2) →(24.1, 60.8)→(32.1, 55.9)→(36.6, 49.2)→(38.9,46.5) →(41.7, 36.9)→(42.6, 35.3)→(45.6, 30.8) →(47.8, 26.5)
	CMOPSO	(11.2, 85.2)→(13.9, 79.9)→(16.1, 75.6)→(19.5, 68.6)→(21.1, 65.3)→(23.1, 61.2)→(24.3,59.2) →(28.1,53.8)→(33.8, 45.8)→(36.3, 42.3)→(38.4, 39.4) →(40.5,36.4)→(43.0, 32.8)→(44.6, 30.6) →(47.8, 26.2)
$STOP2$	Dijkstra's	(9.4, 92,5)→(12.1, 94.1)→(21.5, 94.2)→(25.3, 94.9)→(58.2, 92.5)→(60.3, 90.9)→(59.9, 86.8) →(63.1, 81.1)→(69.3, 84.8)→(75.6, 84.7)→(79.2, 79.8)→(83.5, 71.7)
	CMOPSO	(14.3,93.2)→(20.2, 93.5)→(29.1, 94.1)→(37.9, 93.2)→(50.6, 90.5)→(57.9, 88.1)→(61.6, 86.9) →(62.3, 86.7)→(68.5, 84.6)→(74.8, 82.5)→(80.0, 77.7)→(83.3, 74.7)
$STOP3$	Dijkstra's	(4.6, 92.6)→(2.9, 83.2)→(1.3, 71.1)→(1.3, 21.1)→(1.9, 15.5)→(4.7, 12.3)→(8.1, 6.7)
	CMOPSO	(6.5, 85.8)→(5.0, 71.1)→(2.1, 43.5)→(2.0, 39.4)→(3.0, 30.9)→(4.6, 24.7)→(7.4, 12.9)

in the robot's working environment. In the second level based on the free space found in the first level, Dijkstra's algorithm is applied to obtain the collision-free path from the starting point to the target point. Lastly, a CMOPSO which takes the collision-free path as input, is applied to find the optimal robot path planning. The methodology on the update of the archive, the global best and the accelerated update particle position make the proposed algorithm more effective to reach the optimal solution. The simulation results make a comparison between the pure Dijkstra's algorithm to demonstrate the superiority of the proposed method in terms of the quality solutions. The results definitely demonstrate the ability of the novel hierarchical approach based on CMOPSO to solve the robot path planning problem because it is capable of providing high quality solutions at a very short actual execution time. In the future work, the proposed approach will be implemented on a real robot swarm to evaluate the performance.

References

[1] Vallv, J. Cetto, J. A. (2015). Potential information fields for mobile robot exploration, Robotics and Autonomous Systems, Vol. 69, pp. 68−79.

[2] A. Chevalier, M. V. D. Bossche, C. Copot, R. De Keyser, Formation Control Strategies for Emulation of Field Covering, 18th IEEE International Conference on System Theory, Control and Computing, Sinaia, Romania, pp. 526−531, 2014.

[3] Bac, C. W. Roorda, T. Reshef, R. Berman, S. Hemming,J. Henten, E. J. (2015). Analysis of a motion planning problem for sweet-pepper harvesting in a dense obstacle environment, Biosystems Engineering, pp. 1−13, doi:10.1016/j.biosystemseng.2015.07.004.

[4] Chevalier, A. Copot, C. Cristescu, S. M. Ionescu, C. M. De Keyser, R. (2013). Emulation of a Highway Bottleneck Using Leader-Follower Formation Control, 8th IEEE International Symposium on Applied Computational Intelligence and Informatics, Timisoara, Romania, pp. 131−136.

[5] Hernandez, A. Copot, C. Cerquera, J. D. Murcia, H. De Keyser, R. (2014). Formation control of UGVs using an UAV as remote vision sensor, 19th World Congress: The International Federation of Automatic Control, Proceedings. pp.11872−11877.

[6] Li, Y. Chen, L. Tee, K. P. Li, Q. (2015) Reinforcement learning control for coordinated manipulation of multi-robots, Neurocomputing 170, pp. 168−175.

[7] Lope, D. Maravall, J. D. Quionez, Y. (2015). Self-organizing techniques to improve the decentralized multi-task distribution in multi-robotsystems, Neurocomputing 163, pp. 47−55.

[8] Mac, T. T. Copot,C. Tran, T. D. De Keyser, R. (2016). Heuristic approaches in robot path planning: a survey, Robotics and Autonomous Systems, Vol. 86, pp. 13−28.

[9] Masehian, E. Sedighizadeh, D. (2007) Classic and heuristic approaches in robot motion planning−a chronological review, Proceeding of World Academy of Science, Engineering and Technology 23, pp. 101−106.

[10] Ghita, N. Kloetzer, M. (2012). Trajectory planning for a car−like robot by environment abstraction, Robotics and Autonomous Systems 60, pp. 609−619.

[11] Chiang, H. T. Malone, N. Lesser, K. Oishi, M. Tapia, L. (2015). Path Guided Artificial Potential Fields with Stochastic Reachable Sets for Motion Planning in Highly Dynamic Environments, 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington, pp. 2347−2354.

[12] Nazif, A. N. Davoodi, A. Pasquier, P. (2011). Multi−agent area coverage using a single query road map: A swarm intelligence approach, Advances in Practical Multi−Agent Systems, Studies in Computational Intelligence, vol. 325, pp. 95−112.

[13] Singh, N. N. Chatterjee, A. Rakshit, A. (2011). A two−layered subgoal based mobile robot navigation algorithm with vision system and IR sensors, Emerging Research in Artificial Intelligence and Computational Intelligence, Communications in Computer and Information Science, vol. 237, pp. 325−334.

[14] Duan, H. Huang, L. (2014). Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning, Neurocomputing, Vol. 125, pp. 166−171.

[15] Ster, B. (2004) An integrated learning approach to environment modeling in mobile robot navigation, Neurocomputing 57, pp. 215−238.

[16] Jarrah, R. A. Shahzad, A. Roth, H. (2015). Path Planning and Motion Coordination for Multi-Robots System Using Probabilistic Neuro−Fuzzy, IFAC-Papers OnLine 48-10, pp. 46−51.

[17] Davoodi, M. Panahi, F. Mohades, A. Hashemi, S. N. (2015). Clear and smooth path planning, Applied Soft Computing, vol. 32, pp. 568−579.

[18] Zhang, Y. Gong, D. Zhang, J. (2013). Robot path planning in uncertain environment using multi−objective particle swarm optimization, Neurocomputing 103, pp. 172− 185.

[19] Kaveh, A. Laknejadi, K. (2011). A novel hybrid charge system search and particle swarm optimization method for multi-objective optimization, Expert Systems with Applications 38, pp.15475−15488.

[20] Li, X. Ma, Y. Feng, X. (2013). Self-adaptive autowave pulse-coupled neural network for shortest-path problem, Neurocomputing 115, pp. 63−71.

[21] Zhang, Y. Wu, L. Wei, G. Wang, S. (2011). A novel algorithm for all pairs shortest path problem based on matrix multiplication and pulse coupled neural network, Digital Signal Processing 21, pp. 517−521.

[22] Liu, G. Qiu, Z. Qu, H.Ji, L. (2015). Computing k shortest paths using modified pulse-coupled neural network, Neurocomputing 149, pp.1162−1176.

[23] Hong. T. S., Nakhaeinia, D. Karasfi, B. (2012). Application of fuzzy logic in mobile robot navigation, Fuzzy Logic - Controls, Concepts, Theories and Applications, pp. 21−36.

[24] Lilly, J. H. (2007). Evolution of a negative-rule fuzzy obstacle avoidance controller for an autonomous vehicle, IEEE Trans. Fuzzy Systems, Vol.15, pp. 718−728.

[25] A. Zhu, S. X. Yang, A Fuzzy Logic Approach to Reactive Navigation of Behavior-based Mobile Robots, IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, pp. 5045−5050, 2004.

[26] Hossain, M. A. Ferdous, I.(2015) Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, Robotics and Autonomous Systems, Vol. 64, pp. 137−141.

[27] Mo, H. Xu, L. (2015). Research of biogeography particle swarm optimization for robot path planning, Neurocomputing, Vol. 148, pp. 91−99.

[28] Pop, P. C. Matei, O.Sitar, C. P. (2013). An improved hybrid algorithm for solving the generalized vehicle routing problem, Neurocomputing 109, 76−83.

[29] Ahmed, F. Deb, K. (2013) Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms, Soft Computing, Vol. 17, pp.1283−1299.

[30] Mittal, S. Deb, K. (2007). Three-dimensional offline path planning for UAVs using multi-objective evolutionary algorithms, Proc. IEEE Congress on Evolutionary Computation, Singapore, pp. 3195−3202.

[31] Zhang, Y. Wang, S. Ji, G. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications, Mathematical Problems in Engineering, Volume 2015, Article ID 931256, pp. 1-38.

[32] Panda, S. Padhy, N. P. (2008) Comparison of particle swarm optimization and genetic algorithm for FACTS-based controller design, Applied Soft Computing, Vol. 8, pp. 1418−1427.

[33] Mohemmed, A. W. Sahoo, N. C. Geok, T. K. (2008) Solving shortest path problem using particle swarm optimization, Applied Soft Computing, Vol. 8, pp. 1643−1653.

[34] Hao,Y. Zu, W. Zhao, Y. (2007) Real-time obstacle avoidance method based on polar coordination particle swarm optimization in dynamic environment, in: Proc. 2nd IEEE Conf. Ind. Electron. Appication, pp. 1612−1617.

[35] Xu, W. F. Li, C. Liang, B. Liu, Y. Xu, Y. S. .(2008) The Cartesian path planning of free oating space robot using particle swarm optimization, Int. J. Adv. Rob. Syst, Vol. 5 (3), pp. 301−310.

[36] Gong, D. W. Zhang, J. H. Zhang, Y. (2011). Multi-objective particle swarm optimization for robot path planning in environment with danger, J. Computing Vol. 6 (8), pp. 1554−1561.

[37] Chen, X. Li, Y. (2006) Smooth Path Planning of a Mobile Robot Using Stochastic Particle Swarm Optimization, IEEE International Conference on Mechatronics and Automation, Luoyang, China, pp. 1722− 1727.

[38] Purcaru, C. Precup, R. E. Iercan, D. Fedorovici, L. O. David, R. C. (2013). Hybrid PSOGSA robot path planning algorithm in static environments with danger zones, 17th IEEE International Conference on System Theory, Control and Computing, Sinaia, Romania, pp. 434−439.

[39] Zhang, Y. Jun, Y. Wei, G. Wu, L. (2010) Find multi-objective paths in stochastic networks via chaotic immune PSO, Expert Systems with Applications 37, pp. 1911−1919.

[40] Zhang, Y. Wu, L. Wang, S.(2013). UCAV Path Planning by Fitness-Scaling Adaptive Chaotic Particle Swarm Optimization, Mathematical Problems in Engineering, Volume 2013, pp.1−9.

[41] Lai, X. C. Ge, S. S. (2007) Hierarchical Incremental Path Planning and Situation Dependent Optimized Dynamic Motion Planning Considering Accelerations, IEEE transactions on systems, man, and cybernetics−part B: cybernetics, vol. 37, No. 6, pp. 1514−1554.

[42] Zou, L. Guo, Q. Xu, X. Fu, H. (2015). A hierarchical path planning approach based on A* and least squares policy iteration for mobile robots, Neurocomputing, Vol. 170, pp. 257−266.

[43] Yang, X. Moallem, M. Patel, R. V. (2007). A Layered Goal−Oriented

Fuzzy Motion Planning Strategy for Mobile Robot Navigation, IEEE transactions on systems, man, and cybernetics−part B: cybernetics, vol. 37, No. 6, pp. 1514−1554.

[44] Kala, R. Shukla, A. Tiwari, R. (2011) Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, Neurocomputing, Vol. 74, pp. 2314−2335.

[45] Kaveh, A. (1979). A combinational optimiation problem; optimal generalized cycle bases, Computer methods in applied mechanics and engineering 20, pp. 39-51.

[46] Kaveh, A. Kalatjari, V. (2003). Topology optimization of trusses using genetic algorithm, force method and graph theory, International journal for numerical in engineering, pp. 771−791.

[47] Cagigas, D. (2005). Hierarchical D^* algorithm with materialization of costs for robot path planning, Robotics and Autonomous Systems, Vol.52, pp. 190−208.

[48] Clerc, M. Kennedy, J. (2002). The particle swarm−explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, Vol. 6, pp. 58−73.

[49] Li, Y., Guan, Y. (2012). A Path Planning Method to Robot Soccer Based on Dijkstra Algorithm, Advances in Intelligent and Soft Computing, vol. 149, pp. 89−95.

[50] Coello, C. A. C.Pulido, G. T. Lechuga, M. S. (2004) Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation, Vol. 8(3), pp. 256−279.

[51] Yang, X. S.Deb, S. . Fong, S. (2011). Accelerated Particle Swarm Optimization and Support Vector Machine for Business Optimization and Applications, Networked Digital Technologies, Communications in Computer and Information Science, Vol. 136, Springer, pp. 53−66.

[52] M. Barbehenn, (1998). A Note on the Complexity of Dijkstras Algorithm for Graphs with Weighted Vertices, IEEE transasctions on computers, Vol. 47, No. 2.

[53] Jensen, M. T. (2003). Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II, IEEE transasctions on evolutionary computation, vol.7, no. 5, pp. 503−515.