An Adaptive Multi-Objective Particle Swarm Optimization Algorithm for Multi-Robot Path Planning

Dr. Nizar Hadi Abbas
Department of Electrical Engineering
College of Engineering - University of Baghdad
E-mail: drnizaralmsaodi@gmail.com

Jaafer Ahmed Abdulsaheb
Department of Electrical Engineering
College of Engineering - University of Baghdad
E-mail: jaafer.almadhhachi@gmail.com

ABSTRACT

This paper discusses an optimal path planning algorithm based on an Adaptive Multi-Objective Particle Swarm Optimization Algorithm (AMOPSO) for two case studies. First case, single robot wants to reach a goal in the static environment that contain two obstacles and two danger source. The second one, is improving the ability for five robots to reach the shortest way. The proposed algorithm solves the optimization problems for the first case by finding the minimum distance from initial to goal position and also ensuring that the generated path has a maximum distance from the danger zones. And for the second case, finding the shortest path for every robot and without any collision between them with the shortest time. In order to evaluate the proposed algorithm in term of finding the best solution, six benchmark test functions are used to make a comparison between AMOPSO and the standard MOPSO. The results show that the AMOPSO has a better ability to get away from local optimums with a quickest convergence than the MOPSO. The simulation results using Matlab 2014a, indicate that this methodology is extremely valuable for every robot in multi-robot framework to discover its own particular proper path from the start to the destination position with minimum distance and time.

Key words: multi-robot system, path planning, multi-objective approaches, adaptive multi-objective particle swarm optimization, danger zones.

تطوير خوارزمية اسراب الطيور متعددة الوظائف لتخطيط المسار لآكثر من روبوت

د. نزار هادي عباس
قسم الهندسة الكهربائية
كلية الهندسة، جامعة بغداد

جعفر احمد عبد الصاحب
قسم الهندسة الكهربائية
كلية الهندسة، جامعة بغداد

الخلاصـــة

في هذا البحث تم عرض افضل مسار على اساس خوارزمية اسراب الطيور متعددة الوظائف المعدلة لدراسة حالتين . في الحالة الاولى ، تم تطبيق وظيفتين لتوليد افضل مسار ممكن للروبوت المتحرك في بيئة ثابتة مع تجنب التصادم مع العوائق و مصادر الخطر التي قد تكون موجودة في البيئة والثانية ، لتحسين قدرة البحث لنظام يحتوي على عدد من الروبوتات للحصول على اقصر مسار. الخوارزمية المقترحة تحل المشاكل بطريقة الامثلية عن طريق تقليل دالة الهدف والحصول على المسارات الامثل وبدون تصادم كذلك تقلل من طول الطريق الذي يجب اتباعه بواسطة الروبوت وايضاً تضمن ان المسارات المتولدة على مسافة امنة من مصادر الخطر. بعض دوال الاختبار تم استخدامها للمقارنة بين جودة الخوارزمية المطورة مع

الخوارزيمة الاصلية من ناحية الوصول الى افضل الحلول. النتائج اظهرت ان الخوارزمية المطورة تمتلك قدرة افضل للخروج من الامثلية المحلية والحصول على تقارب اسرع من الخوارزمية الاصلية . نتائج المحاكاة ونتائج التحقق تبين بأن هذه المنهجية هي قيمة للغاية لكل روبوت في إطار منظومة متعددة الروبوتات لاكتشاف الطريق الصحيح الخاص به من موقع البداية الى الهدف مع الحصول على الحد الأدنى من المسافة المقطوعة والزمن.

الكلمات الرئيسية: نظام متعدد الروبوت، تخطيط المسار، طرق الوظائف المتعددة ، خوارزمية اسراب الطيور متعددة الوظائف ، مناطق الخطر.

1. INTRODUCTION

The Multi Robot System (MRS) can be described as a group of robots working in the same environment. However, robotic systems may range from simple sensors, acquiring and processing data, to complex human like machines, able to interact with the environment in fairly complex ways. MRS has been widely applied to rescue, industrial, exploration of outer space areas, due to its characteristics of reliability, robustness, and economy. Path planning has been known as one of the main problems in the MRS. The objective was to choose the optimum path without collision among them in the specified arena, **Li, et al., 2009**.

One of the most important tasks in the moving of mobile robots in static (fixed) environments in the existence of multi-obstacles was to arrive the goal as fast as possible using an optimal trajectory. Sometimes, the environments that the robots are working on may be included danger zones (sensitive areas), so it must be considered in the robot path planning (RPP) algorithms. In addition to generating the shortest path, the RPP algorithms also generate trajectories at safe spaces from the danger zones in the arena, **Gong, et al., 2011**.

A MOPSO algorithm is utilized in, **Zhang, et al., 2013,** to generate trajectories for mobile robots that are working on the environments that the robots are working on and may be included danger sources. R. Kala introduced a co-evolutionary genetic programming algorithm to produce a comprehensive, optimal path for multi-robots in map of the maze, where every robot searching for the shortest path in addition to that, each robot avoids the collision with other robots. The results show that this method can find the best paths for whatever number of mobile robots in various arenas and scenarios, **Kala, 2012**. **Kaluder, et al.,2011,** showed a modified Asano's algorithm which is performed to locate the visibility polygons and graphs. According to the result, a cubical complexity of the algorithm is shown, based on reflection point numbers.

The methods that are used for solving RPP can be separated into two methods: traditional and intelligent, the first method contains visibility graphs, configuration space (C-Space) and artificial potential field.**, Jaillet, et al., 2010,** addressed path planning issue to consider the cost function defined over the C-Space. The proposed design computes minimum cost paths that follow the valleys and saddle points of the C-Space cost map. According to the results this method is very effective. The second method is a group of obstacles and points in the Euclidean plane and it was applied to find the Euclidean shortest paths between a group of point (polygonal point). The artificial intelligent path planning includes Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Genetic Algorithm (GA) and so on. **Chung and Xu**,**2010,** suggested a generalized three dimensional (3D) path planning technique for robots using GA with an adaptive evolution process. According to nearest neighbor, the authors presented a new operator, named it a Bind-NN that is randomly separated and reconnect an elitist chromosome. According to the results the efficiency and search effectiveness has a better improvement.

Chakraborty, et al., 2008, presented an alternative method for a co-operative multiple RPP problem using parallel differential evolution algorithms. According to the experimental results this method is very useful for each robot in multi-robot systems for searching and finding the shortest path to target.

In this paper, two case studies based on an Adaptive Multi-Objective Particle Swarm Optimization Algorithm (AMOPSO) for solving RPP problem are presented the first one for single robot and the second for multi-robot. In the first case, the AMOPSO algorithm generates optimal paths by maximizing the distance between the generated paths and the danger zones that exist in the arena and also minimizing the length of the path that needs by the mobile robot to reach the target. In the second case, the AMOPSO algorithm improves the ability for multiple robot system to reach the shortest way.

The rest of this paper is organized as follows: section 2 describes a problem formulation; Section 3 describes the theoretical background; Section 4 describes the proposed AMOPSO algorithm and the simulation results and discussion are presented in section 5. Finally, section 6 gives the research conclusions.

2. PROBLEM FORMULATION

2.1 Single Robot with Danger Zones

The multiple objective optimization algorithm is considered as AMOPSO algorithm, which depends on two objectives to find the optimal paths for single mobile robots in fixed environments that contain a number of obstacles and danger source.

PSO algorithm using a number of the population P of particles that are taking as a random position in the work space. Every particle has a velocity (randomized) assigned to which they have changed its position in the work space.

Let (R) be the robot that is used in a known environment in the task of finding the optimal path with existing of danger zones and multi-obstacles. A population of particles (P) contains a number of particles (N), are participating in the production of optimal path for the robot R.

The static environment includes a group of obstacles {Ok | k =1...p} and group of danger zones {DZk | k =1...r} that are known in advance. As described in, **Gong, et al., 2011,** the danger zone can be found in the environment when the robot is moving to reach the target.

The first objective in this case is to reach the maximum distance among each particle and the danger zones. This means, at each iteration the AMOPSO algorithm attempt to find the maximum Euclidean distance between the particle positions and every point for danger zones. To achieve that, the following objective function is used:

$$f1_i\,(t) = \frac{1}{\sum_{j=1\; j\neq i}^{r} \sqrt{(X_{DZj} - X_i(t))^2 + (Y_{DZj} - Y_i\,(t))^2}}\,,\; i = 1\ldots N, \qquad (1)$$

where (X_{DZj}, Y_{DZj}) are axis for the danger zone's position. When the AMOPSO algorithm reaches the maximum number of iterations, then it eliminated agent’s trajectories which have any collision points with any points of danger zones.

The second objective is to find the minimum distance that is needed by the robot from the start position (X_i(t), Y_i(t)) to the goal point (X_f, Y_f). The objective function which is used to reach the minimum Euclidean distance between the agent current location and the goal is formulated as,

$$f2_i\,(t) = \sqrt{(X_i(t) - X_f)^2 + (Y_i\,(t) - Y_f\,)^2}\ \ , i = 1\ldots N. \tag{2}$$

Eq. (1) and Eq. (2) are brought together into a single objective function:

$$f_i(t) = f1_i\,(t) + f2_i\,(t) \tag{3}$$

Therefore, every iteration finds the maximum distance between the path points and the danger zones points in the arena and also obtain the minimum distance that is needed by the mobile robot.

2.2 Multi-Robot System with Shortest Way

There are three principles used to organize the robot movement in order to reach the goal position without collision with obstacles or other robot in the arena, these principles are:

1) At first, the robot identifies the next position so as to align itself to a goal.
2) This alignment may cause a collision with another robot. This may be happening in the case of more than one robot try to take the same position. Also the collision may be happening with obstacles that found in the next position. To avoid such collision, the robot has to turn left or right by changing its position by increasing x-axis and y-axis with threshold.
3) Finally, if the robot can align itself to the goal without any collision with other robot or obstacles, it will move to next position.

The objective is to minimize the distance that is needed for each robot from the start position to its goal with minimum time. The objective function that is used to minimize the Euclidian distance between the agent current position and the goal point is formulated as:

$$f_i\,(t) = \sqrt{(X_i(t) - X_f)^2 + (Y_i\,(t) - Y_f\,)^2}\ \ , i = 1\ldots N. \tag{4}$$

where, (X_i(t), Y_i(t)) is initial position and (X_f, Y_f) is goal point.

3. THEORETICAL BACKGROUND

3.1 Path Planning

The field of robot path planning (RPP) was begun in 1960's. The RPP problem is a very defy challenging in the field of robotics. The main objective is to find a collision free path from an initial position to a destination position. Robot Navigation (RN) problem has to be concerned with three main matters: accuracy, safety and efficiency. The accuracy and safety issues deal with finding a collision-free path and following the exactly addressed path. Efficiency means that the algorithm searching the shortest distance with acceptable time by not letting the robot to stop and turn many times or take needless steps, which results in squandering of time and energy consumption. **Ehlert, 1999**.

Depending on the environment where the robot located in; RPP can be classified into two types:

1) RPP in static environment: if there are fixed obstacles in the arena.
2) RPP in dynamic environment: if the arena has both fixed and moving obstacles.

Each of these two types could be further subdivided into a sub-group:

1) Global Path Planning (GPP): a total information about fixed obstacles and a path of moving obstacles are known in advance; thus, the GPP can be planned before the robot starts to move (offline).
2) Local Path Planning (LPP): a total information about the environment is not obtainable in advance. So, while it moves through the environment the mobile robot obtains information through sensors (online), **Miao, 2009**.

3.2 Optimization Technique

3.2.1 Standard PSO Algorithm

Particle Swarm Optimization (PSO) algorithm was invented by Kennedy and Eberhart in 1995 is an evolutionary algorithm inspired by the motion of the fish school or bird flocks in nature. It is used in optimizing the continuous nonlinear functions. PSO uses a population of particles (agents) that are moving in the work space and at each iteration a particle memorized the coordinates of the position in the work space associated with better fitness value achieved so far. PSO also stores the position of the best value from the whole particles, **Kennedy and Eberhart,1995**.

A position vector $X_i(t)$ and velocity vector $V_i(t)$ for every particle in the population is**, Rao, 2009**:

$$V_i(t) = (V_i^X(t), V_i^Y(t)),\ i=1...N, \tag{5}$$

$$X_i(t) = (X_i^X(t), X_i^Y(t)),\ i=1...N. \tag{6}$$

where the super-scripts X & Y highlight the vector components in the 2-D work space where the robot is moving. Every agent maintains the best own positions that are reached in the best position vector $P_{i,\, best}$ (t):

$$P_{i,\, best}\ (t) =(P_i^X(t), P_i^Y(t)),\ i=1...N. \tag{7}$$

The best positions of the whole population are maintained in the vector $P_{g,\, best}$ (t):

$$P_{g,\, best}\ (t) =(P_g^X(t), P_g^Y(t)),\ i =1...N. \tag{8}$$

The updating velocity, which is used by the particle in the moving in search space can be found according to:

$$V_i^X (t+1) =w(t)\ V_i^X(t) + c1\ r1\ [P_i^X(t) - X_i^X(t)] + c2\ r2\ [P_g^X(t)-X_i^X(t)],\ i=1...N. \tag{9}$$

$$V_i^Y(t+1) = w(t)\, V_i^Y(t) + c1\, r1\, [P_i^Y(t) - X_i^Y(t)] + c2\, r2\, [P_g^Y(t) - X_i^Y(t)],\ i=1...N. \quad (10)$$

where r1 and r are uniformly distributed random variables in the range of [0,1] , $0.4 \leq w(t) \leq 0.9$ is the inertia weight, c1 and c2 are the weighting factors of the stochastic accelerations pulling the agents towards their final positions and it rang is [0,4]. The next particle position in the search space is obtained using the following equations:

$$X_i(t+1) = X_i(t) + V_i^X(t+1). \quad (11)$$

$$Y_i(t+1) = Y_i(t) + V_i^Y(t+1). \quad (12)$$

Pseudocode that shows the steps of the standard PSO algorithm works illustrated in **Fig. 1**.

3.2.2 MOPSO Algorithm

Mostly, a single condition used by researchers to generate an optimal path, such as the time required by a mobile robot to reach the target or minimum path length. But, in practice, several conditions must be meets to make the path feasible, such as safety, energy consumption, smoothness, etc.

An optimal path for single criterion does not mean that all the other criteria are satisfied, **Fujimura, 1996,** As an example, an energy consumption dose not desired at the expense of shortest path along the path.

So, in multi-objective PSO (MOPSO) algorithm the target is to find a set of different solutions by modifying the original scheme. Three main concepts must be considered when extending PSO to MOPSO. These concepts are **Dehuri, et al., 2008**:

- How to choose the leaders from particles with a view to give priority to non-dominated solutions over those that are dominated?
- How to keep a non-dominated solution that be found within the search process with a view to give a report solution that is non-dominated with respect to all the past agents and not only with respect to the current one?
- How to save the diversity in the swarm with a view to avoid convergence for a single local solution?

Pseudocode that shows the procedure of the standard MOPSO algorithm works presented in **Fig. 2**. Italics are used in **Fig. 2** to clarify the difference process between MOPSO and PSO algorithms.

3.2.3 Multi-objective approaches

When the optimization problem contains more than one objective function, the mission of finding one optimal solutions or more is known as multiple objective optimization. The common methods that are used to deal with multiple objective optimization are: weighted sum and pareto front

1. Weighted Sum Approach

The weighted sum method combines all multiple objective functions into one scalar, composite objective function using the weighted sum, **Yang, 2014**.

$$f(x) = \sum_{m=1}^{M} W_m\, fm(x). \tag{13}$$

The important matter in specifying the weighting coefficient, $W = (W_1, W_2, \ldots, W_m)$ because the strongly solution depends on the selection of W. Obviously, these weights have been positive, satisfying $\sum_{m=1}^{M} W_m = 1$, $W_m \in [0,1]$.

2. Pareto Dominance and Pareto Optimality

In a pareto set, a solution back to the pareto set, if there is no other solution can improve at least one objective without degrading on any other one from the objective. In the context of Multi-Objective Optimization (MOO), formally, a decision vector $\vec{u} \in \Omega$ is said to pareto dominate vector $\vec{v} \in \Omega$, in a minimization context, if and only if:
111111

$$\forall i \in \{1, \mathrm{K}, N\}\, f_i(\vec{u}) \le f_i(\vec{v}),$$

$$and\ \exists_j \in \{1, \mathrm{K}, N\}, f_i(\vec{u}) < f_i(\vec{v}). \tag{14}$$

In the context of Multi-Objective Optimization MOO, Pareto dominance is used to compare and rank decision vectors: $\vec{u}$ dominating $\vec{v}$ in the Pareto sense means that $\vec{\mathrm{F}}(\vec{u})$ is either the same or better than $\vec{\mathrm{F}}(\vec{v})$ for all objectives, and there is at least one objective function for which $\vec{\mathrm{F}}(\vec{u})$ is strictly better than $\vec{\mathrm{F}}(\vec{v})$ **Yang, 2014**.

4. PROPOSED AMOPSO ALGORTHIM

In this paper, an adaptive approach is proposed to adjust the particles velocity and position to overcome the slow convergence problem that emerged in Standard PSO (SPSO) algorithm.

Thus, in the APSO, the particle position is updated such that the highly fitted particle moves slowly when compared to the lowly fitted particle. Therefore, in order to achieve the promising updating, the following particles' parameters should be adapted according to their objective function values, **Humadi, et al., 2013**:

1. The adaptive inertia weight factor (AIWF) w_i^t, is proposed to find out a compromised AIWF that satisfies both exploration (global search) and exploitation (local search). The AIWF is determined as in Eqs. (15 & 16).
2. The adaptive acceleration coefficients (AACs) $c^t_{1,i}$ and $c^t_{2,i}$ they are used to award the efficient particle that has high fitness and punishes the not competent one. These AACs are formulated as in Eqs. (17, 18, 19, 20, 21 & 22).
3. The adaptive random numbers (ARNs) $r^t_{1,i}$ and $r^t_{2,i}$, are proposed to increase the movement impact on the third term (swarm) and decrease the movement influence on the second term (individual) of Eqs. (9) and (10). These ARNs are written as in Eqs. (23&24).

$$W_i = (W_{max} - W_{min}) * (e^{-(\frac{Max\,(Pbesti)}{Gbest})^{\wedge}2} * log_{Tmax}(T_i + 1)) + W_{min}. \tag{15}$$

$$W_i = (W_{max} - W_{min}) * L * log_{Tmax}(T_i + 1) + W_{min}. \tag{16}$$

C_1 decrease from 1.5 ~ 0.5, C_2 increase from 1.5 ~ 2.5

$$C_1 = Z + L * \frac{Ti}{Tmax} \quad (17)$$

$$C_2 = Z + L * \frac{Ti}{Tmax}. \quad (18)$$

C_1 decrease from 1.5 ~ 0.5, C_2 decrease from 1.5 ~ 0.5

$$C_1 = Z - L * \frac{Ti}{Tmax}. \quad (19)$$

$$C_2 = Z - L * \frac{Ti}{Tmax} \quad (20)$$

C_1 decrease from 1.5 ~ 0.5, C_2 increase from 1.5 ~ 2.5

$$C_1 = Z - L * \frac{Ti}{Tmax} . \quad (21)$$

$$C_2 = Z + L * \frac{Ti}{Tmax}. \quad (22)$$

$$r_1 = rand_1 * L * \frac{Ti}{Tmax}. \quad (23)$$

$$r_2 = rand_2 * L * \frac{Ti}{Tmax}. \quad (24)$$

where, W_i must be between (0.4 and 0.9)

- 0.4 when the first part of the Eqs. (15 & 16) equal to zero
- 0.9 when the ($e^{-(\frac{Max(Pbesti)}{Gbest})^2} * \log_{Tmax}(T_i + 1)$) equal to one
- 0.9 when the $L * \log_{Tmax}(T_i + 1)$ equal to one
- 0.4 ~ 0.9 when ($e^{-(\frac{Max(Pbesti)}{Gbest})^2} * \log_{Tmax}(T_i + 1)$) not equal to zero or one
- 0.4 ~ 0.9 when $L * \log_{Tmax}(T_i + 1)$ not equal to zero or one

Z = Constant = 1.5

T_i = Current iteration
T_{max} = Maximum number of iterations
L must be in this range $0 \le L \le 1$

- $M = \frac{1}{N} * \sum_{i=1}^{n} Pbesti$
- $V^2 = \frac{1}{N-1} * \sum_{i=1}^{n} (Pbesti - M)^2$
- $Y = \frac{Pbesti - M}{V}$
- $L = e^{-|Y|}$

$rand_1$ & $rand_2$ = random number between (0 ~ 1)
So, in this paper four AMOPSO cases are simulated, these cases are listed in **Table 1.**

5. SIMULATION RESULTS AND DISCUSSION

5.1 Simulation Parameter Settings

The following parameters of the AMOPSO path planning algorithm have been used in the experiment: N = 8, maximum number of iterations t_{max}=120, initial velocities v_x^i = random and v_y^i = random, acceleration constants $0.5 \le c_1 \le 2.5$ and $0.5 \le c_2 \le 2.5$ and inertia weight $w_{min} \le w \le w_{max}$, $w_{min} = 0.4$ and $w_{max} = 0.9$.

The Matlab2014a programming language used to implement the simulation code for path planning and executing on the system with 2.60GHz CPU and 2.0G RAM.

5.2 Benchmark Test Functions

Several unimodal and multimodal benchmark functions have been adopted from, **Vesterstrom and Thomsen, 2004**. The list of the test functions and some of their characteristics can be seen in **Table 2** and **Table 3**. In the **Table 2**, the “Range” column gives the defined range of the parameter and the “dim” column shows the number of dimensions used for each function and in the **Table 3** “Min. & Av.” column, the first value represents the minimum of optimal solution and the second one represents the average of optimal solution that obtained over 100 runs are given. Functions f1 ~ f3 are unimodal and f4 ~ f6 multimodal.

The function f1 is the Sphere function:

$$f1(x) = \sum_{i=1}^{D} x_i^2. \tag{25}$$

The function f2 is the Quadric function:

$$f2(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{D} x_j^2\right)^2. \tag{26}$$

The function f3 is the Quadric Noise function:

$$f3(x) = \sum_{i=1}^{D} i x_i^4 + random\ (0,1). \tag{27}$$

The function f4 is the Rastrigin function:

$$f4(x) = \sum_{i=1}^{D} [x_i^2 - 10 \cos (2\pi x_i) + 10]. \tag{28}$$

The function f5 is the Ackley function:

$$f5(x) = -20\, exp(-0.2\sqrt{\frac{1}{D\sum_{i=1}^{D} x_i 2}}) - exp\left(\frac{1}{D \cos \sum_{i=1}^{D} (2\pi x_i)}\right) + 20 + e. \tag{29}$$

The function f6 is the Griewank function:

$$f6(x) = 1/4000 \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos (x_i/\sqrt{i}\,) + 1. \tag{30}$$

The initialization equation for x in the range of search space:

$$x = (b - a) * random\ (0,1) + a. \tag{31}$$

where a is the maximum limit in the search space and b is the minimum limit in the search space.

In SPSO, $c_1=c_2=2$, r_1=random (0,1), r_2=random (0,1), w_{min} = 0.4 and w_{max} = 0.9, the inertia weight equation, **Rao, 2009**:

$$w(i)= w_{max} -(\frac{w_{max}- w_{min}}{i_{max}}) * i \tag{32}$$

5.3 Simulation Results and Discussion

5.3.1 Case study 1: environment with 2 obstacles and 2 danger zones

In this section, the environment used for the planning is a 11*11 meter, all obstacle positions are listed in **Table 4** and result in **Table 5.** Starting point is (0,0) and target point is (10,10). The experiment has achieved a feasible solution; the best trajectory achieved by AMOPSO1 is illustrated in **Fig. 3**, best trajectory achieved by AMOPSO2 is shown in **Fig. 4**, best trajectory achieved by AMOPSO3 is depicted in **Fig. 5**. While the best trajectory achieved by AMOPSO4 is shown in **Fig. 6**. A best distance achieved by Hybrid PSO-GSA in, Purcaru, et al., 2013, is illustrated in **Fig. 7**. By comparing the results achieved in **Figs. 3, 4, 5, 6** and **Fig. 7** the AMOPSO1, AMOPSO2, AMOPSO3 and AMOPSO4 has a maximum distance from the danger zone and minimum length to reach the target than Hybrid PSO-GSA, according to this if multi robot used in the arena the AMOPSO give better results than Hybrid PSO-GSA. The results obtained from Pareto are better than the results obtained from weighted sum. In pareto, the average of maximum distance from danger zone is 40.04 and the average for minimum path is 14.72 while in weighted sum, the average of maximum distance from danger zone is 35.1 and the average for minimum path is 14.44.

5.3.2 Case study 2: Multi-robot in environment with 5 obstacles

In this section, the environment used for the planning is a10*10 meters, all results are listed in **Table 6.** The experiment has achieved a feasible solution; the global path was achieved by AMOPSO is illustrated in **Fig. 8**, In the graph, the start point of robot 1 is [0,0] and the stop point is [10, 10], the start point of robot 2 is [2.5, 0] and the stop point is [2.5, 10], the start point of robot 3 is [7.5, 0] and the stop point is [2.5, 10], the start point of robot 4 is [0, 2.5] and the stop point is [7.5, 10], the start point of robot 5 is [0, 7.5] and the stop point is [10,0]. Best distances achieved by Immune Ant Colony Optimization Network Algorithm in, **Hao, and Xu, 2014,** are listed in **Table 7.** By comparing the results achieved in **Tables 6** and **7** the AMOPSO1 has a minimum time to reach the target than Immune Ant Colony Optimization Network Algorithm and pareto reach the target in 44.8 times less than weighted sum and 562 times less than Immune Ant Colony Optimization Network Algorithm.

6. CONCLUSION

In this paper, two case studies for path planning model based on AMOPSO is developed to enhance the performance of the multi-robot path planning. In the first case, the algorithm generates an optimal collision free trajectory in static environments that can contain known multiple obstacles and multiple danger zones and the second case improve the ability of multiple robot system to reach the shortest way. The algorithm achieved by Matlab 2014a and has been applied on two maps, first map, including two obstacles and two danger zones and the second including different barriers. The result of first case shows that the AMOPSO generates an optimal path by maximizing the distance between the generated paths and the danger zones that exist in the environment and also minimizing the length of the path that needs by the mobile robot to reach the target, these results are better than the results achieved by Hybrid PSO-GSA. And the result for the second one shows that AMOPSO could be suit for a multi-robot system to find the shortest path and without collision between them. These results show that the AMOPSO has shortest time compared with time achieved by Immune Ant Colony Optimization Network Algorithm.

REFRENCES

- Chakraborty, J., Konar, A., Chakraborty and U. K. and Jain, L. C., 2008, *Distributed Cooperative Multi-robot Path Planning using Differential Evolution*, In Proceeding of IEEE World Congress on Computational Intelligence, Hong Kong, pp.718-725.

- Chung, W. K., and Xu, Y., 2010, *A Generalized 3-D Path Planning Method for Robots Using Genetic Algorithm with an Adaptive Evolution Process*, In Proceeding of 8th World Congress on Intelligent Control and Automation, Jinan, China, pp.1354-1360.

- Dehuri, S., Ghosh, A., and Cho, S-B., 2008, *Particle Swarm Optimised Polynomial Neural Network for Classification: A Multi-Objective View*, International Journal of Intelligent Defence Support Systems, vol. 1, no. 3, pp.225-253.

- Ehlert, P. A. M., 1999, *The Use of Artificial Intelligence Robots*, Report on research project, Delft University of Technology, Netherlands.

- Fujimura, K., 1996, *Path Planning with Multiple Objectives*, IEEE Robotics and Automation Magazine, vol. 3, no.1, pp. 33-38.

- Gong, D. W., Zhang, J. H., and Zhang, Y., 2011, *Multi-Objective Particle Swarm Optimization for Robot Path Planning in Environment with Danger Sources*, Journal of Computers, vol. 6, no. 8, pp. 1554–1561.

- Hao, W., and XU, X., 2014, *Immune Ant Colony Optimization Network Algorithm for Multi-Robot Path Planning*, In Proceedings of the 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, pp. 1118-1121.

- Humadi, R. A., Abbas, N. H., and Hammadi, W., 2013, *PID Parameters Optimization Using Adaptive PSO Algorithm for a DCSM Position Control*, International Journal of Electrical Engineering and Technology, vol. 4, Issue 4, pp. 1-13.

- Jaillet, L., Cortés, J., and Siméon, T., 2010, *Sampling-based Path Planning on Configuration-Space Costmaps*, IEEE Transactions on Robotics, vol. 26, no. 4, pp.635-646.

- Kala, R., 2012, *Multi-robot Path Planning Using Co-evolutionary Genetic Programming*, Expert Systems with Applications, vol.39, no. 3, pp.3817-3831.

- Kaluder, H., Brezak, M., and Petrovic, I., 2011, *A Visibility Graph based Method for Path Planning in Dynamic Environments*, In Proceeding of 34th International Convention on MIPRO, Opatija , Croatia, pp.717-721.

- Kennedy, J., and Eberhart, R. C., 1995, *Particle Swarm Optimization*, In Proceedings of IEEE International Conference Neural Networks, NJ, USA, pp. 1942–1948.

- Li, H., Yang, S. X., and Seto, M. L., 2009, *Neural Network based Path Planning for a Multirobot System with Moving Obstacles*, In IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 39, no. 4, pp. 410-419.

- Miao, H., 2009, *Robot Path Planning in Dynamic Environments Using a Simulated Annealing Based Approach*, Master thesis, Queensland University of Technology, Queensland, Australia.

- Purcaru, C., Precup, R.-E., Iercan, D., Fedorovici, L.O., and David, R.-C., 2013, *Hybrid PSO-GSA Robot Path Planning Algorithm in Static Environments with Danger Zones*, In Proceedings of the 17th International Conference on System Theory, Control and Computing, Sinaia, Romania, pp. 434-439.

- Rao, S. S., 2009, "*Engineering Optimization: Theory and Practice*", 4th Edition, Wiley.

- Vesterstrom, J., and Thomsen, R., 2004, *A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems*, IEEE Congress on Evolutionary Computation, Portland, USA, vol.2, pp. 1980-1987.

- Yang, X., 2014, *Nature-Inspired Optimization Algorithms*, 1st Edition, Elsevier.

- Zhang, Y., Gong, D.-W., and Zhang, J.-H., 2013, *Robot Path Planning in Uncertain Environment Using Multi-Objective Particle Swarm Optimization*, Neurocomput., vol. 103, pp. 172–185.

```
Begin
  for each particle in the swarm
    Initialize its position & velocity randomly
  end for
do
  for each particle in the swarm
    Evaluate the fitness function
    if the objective fitness value is better than the personal best objective fitness value
(P_best) in history, current fitness value set as the new personal best (P_best)
    end if
  end for
  From all the particles or neighbourhood, choose the particle with best fitness value as
the G_best
  for each particle in the swarm
    Update the particle velocity according to Eqs. 9 & 10
    Update the particle position according to Eqs. 11 & 12
  end for
 until stopping criteria is satisfied
 end begin
```

Figure 1. Pseudocode of the standard PSO algorithm.

```
Begin
  for each particle in the swarm
    Initialize particles position & velocity randomly
  end for
    Initialize External Archive (EA) (initially EA empty)
    Quality (leader)
do
  for each particle in the swarm
    select a particle (leader) from EA
    Evaluate the fitness function
    if the objective fitness value is better than the personal best objective fitness value
(P_best) in history then
      current fitness value of the objective function is set as the new P_best
    end if
    Update the particle velocity according to Eqs. 9 & 10
    Update the particle position according to Eqs. 11 & 12
  end for
    Update leader in EA
    Quality (leader)
until stopping criteria is satisfied
report the results of EA
end begin
```

Figure 2. Pseudocode of the standard MOPSO algorithm.

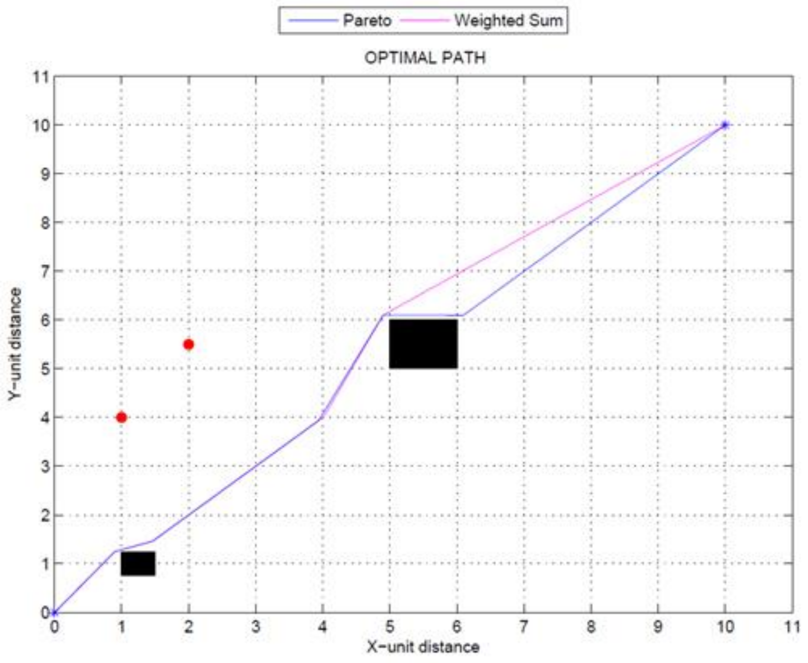

Figure 3. Best results achieved by AMOPSO1.

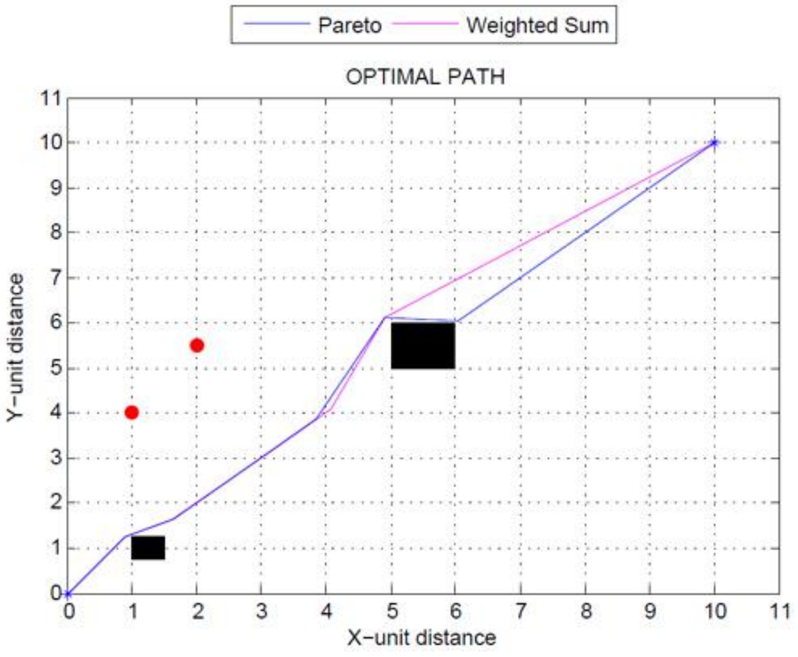

Figure 4. Best results achieved by AMOPSO2.

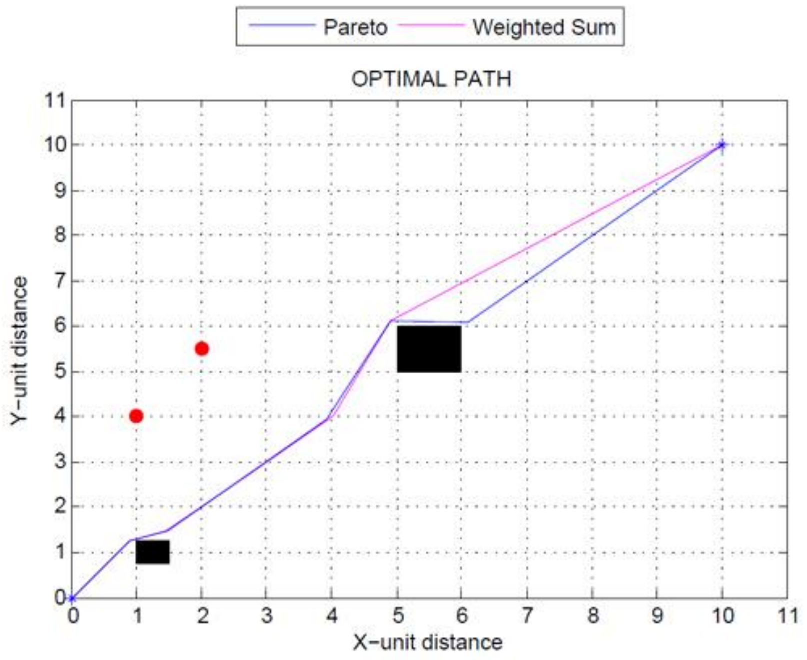

Figure 5. Best results achieved by AMOPSO3.

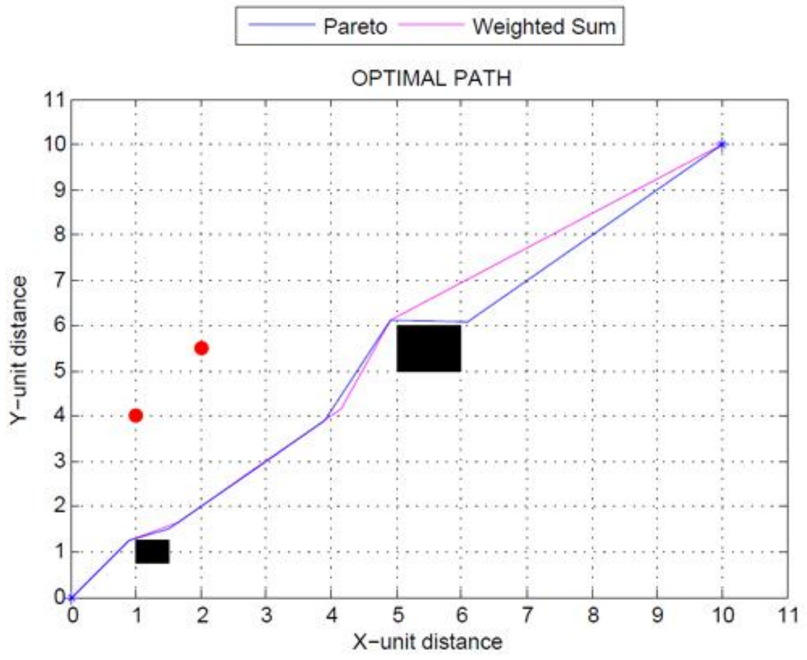

Figure 6. Best results achieved by AMOPSO4.

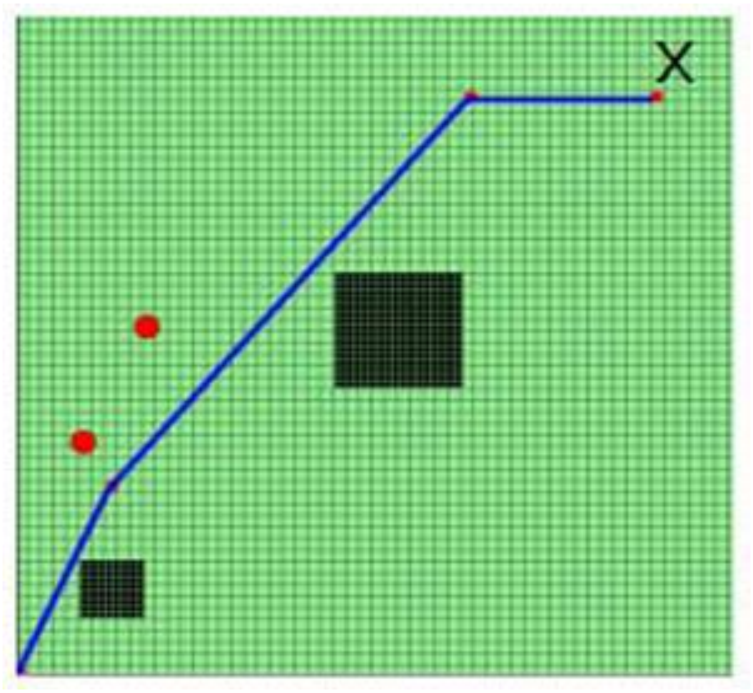

Figure 7. Best results achieved by Hybrid PSO-GSA, **Purcaru, et al., 2013**.

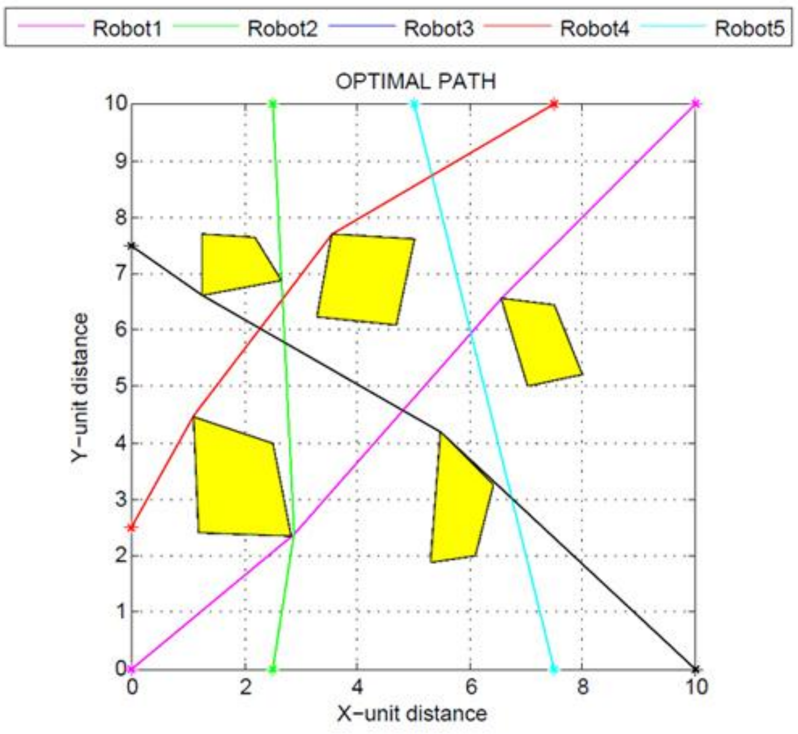

Figure 8. Best results achieved for case study 2.

Table1. AMOPSO cases

Name	W Equation	C1 & C2 Equation	r1 & r2 Equation
AMOPSO1	15	17 & 18	23 & 24
AMOPSO2	15	19 & 20	23 & 24
AMOPSO3	16	19 & 20	23 & 24
AMOPSO4	16	21 & 22	23 & 24

Table 2. Different benchmark functions on the SPSO and AMOPSO have been tested.

Benchmark fun.	Dim	Range
f1	30	$[-100,100]^D$
f2	30	$[-100,100]^D$
f3	30	$[-1.28,1.28]^D$
f4	30	$[-5.12,5.12]^D$
f5	30	$[-32,32]^D$
f6	30	$[-600,600]^D$

Table 3. Results for benchmark functions based on the SPSO and AMOPSO1, AMOPSO2, AMOPSO3 and AMOPSO4 algorithms.

f	SPSO	AMOPSO1	AMOPSO2	AMOPSO3	AMOPSO4
	Min. & Av.	**Min. & Av.**	**Min. & Av.**	**Min. & Av.**	**Min. & Av.**
f1	0.16239 0.26797	0.020316 0.020322	0.022755 0.02276	0.020315 0.020318	0.021744 0.021748
f2	0.61426 0.71556	0.063806 0.063186	0.066616 0.066639	0.060719 0.060745	0.06022 0.060259
f3	0.20384 8.4377	0.07885 0.087416	0.065096 0.070362	0.072637 0.081607	0.031205 0.048147
f4	0.16376 2.1796	0.044989 0.080379	0.078139 0.09372	0.020808 0.02131	0.019634 0.021526
f5	0.21717 0.30033	0.018962 0.019161	0.020767 0.020981	0.021648 0.02187	0.021819 0.022042
f6	0.20413 0.36459	0.027966 0.028254	0.027475 0.0281	0.028591 0.028887	0.028807 0.029154

Table 4. Coordinates of obstacles and danger zone for the case study 1.

Obstacles	**Center Position**
1	1.2,1
2	5.5,5.5
Danger zone	**Position**
1	1,4
2	2,5.5

Table 5. Result for case study 1.

Algorithm	**Max. distance from danger zone**		**Min. distance from start to target position**	
	Pareto	**Weighted Sum**	**Pareto**	**Weighted Sum**
AMOPSO1	46.251	39.963	14.725	14.437
AMOPSO2	38.472	33.893	14.72	14.431
AMOPSO3	37.867	33.38	14.729	14.439
AMOPSO4	37.576	33.165	14.724	14.436

Table 6. Result for case study 2.

Robot number	**Min. distance from start to target position**		**Time**	
	Pareto	**Weighted Sum**	**Pareto**	**Weighted Sum**
Robot 1	14.7	14.7	0.0011602	0.051955
Robot 2	10.041	10.041		
Robot 3	10.308	10.308		
Robot 4	10.889	10.889		
Robot 5	12.571	12.571		

Table 7. Result for Immune Ant Colony Optimization Network Algorithm, **Hao, and Xu, 2014**.

Robot number	**Min. distance from start to target position**	**Time**
Robot 1	14.1728	0.6525
Robot 2	10.0423	
Robot 3	10.3078	
Robot 4	10.9045	
Robot 5	12.5720	