An Adaptive Multi-Objective Artificial Bee Colony Algorithm for Multi-Robot Path Planning

Asst. Prof. Nizar Hadi Abbas
Jaafar Ahmed Abdulsaheb
University of Baghdad
Electrical Engineering\ Iraq

Abstract

This paper discusses an optimal path planning algorithm based on an Adaptive Multi Objective Artificial Bee Colony Algorithm (AMOABC) for three case studies. First case, two robots wants to reach the different target with two objectives; first objective is to find the minimum distance that is needed by the robot from the start position to the target while the second objective is to find the maximum distance between the paths of the two robots. The second case is to find the optimal path with shortest and smoothest path for the three and four robot. The last one, finding the shortest path for five robots without any collision between them with smoothest and shortest time. The results show that the AMOABC has a better ability to get away from local optimums with a quickest convergence than MOABC. 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.

Keywords: Multi-Robot System, Path Planning, Multi-Objective Approaches, Adaptive Multi-Objective Artificial Bee Colony.

1. Introduction

Multi Robot Systems (MRS) can be described as a group of robots working in the same environments. However, the range of robotic systems starts from simple sensors, processing and acquiring data, to complex humans such as machines, that are able to interreact with environments in fairly complex ways. Multi-robot systems have been widely applied in rescuing, industry, exploration of outer space areas, due to their characteristics of reliability, robustness, and economy. Path planning has been one of the main problems in MRS. The objective of this paper is to choose the optimum path for MRS without collision among them in a specified arena [1].

The Artificial Bee Colony (ABC) algorithms are utilized for solving hard optimization problems, including Robot Path Planning (RPP). ABC fast convergence, strong robustness, and

high flexibility. Also, ABC is suitable for solving multimodal and multidimensional optimization problem. ABC method suffers from local optimizing and poor convergence rate [2].

Bhattacharjee et al. [3] proposed an alternative approach for multi mobile robots path planning by using ABC algorithm. The proposed algorithm minimizes the path length for each robot from predefined initial positions to destination. Agarwal and Goel [4] introduced a new method to solve the problem of mobile robot path planning based on the bee colony algorithm. The proposed algorithm includes two steps: the creation of an initial collision free path from starting point to the target, and the use of the bee colony algorithm to find the optimal initial path. Purcaru et al. [5] proposed a new optimal path planning algorithm based on hybridization between a Particle Swarm Optimization (PSO) and a Gravitational Search Algorithm (GSA). The hybrid PSO-GSA generates optimal collision-free paths by minimizing the length of the path for each robot from its initial position to the target and by maximizing the distance between the robot's path and the other robot's path on the (X, Y) axis.

Masehian and Sedighizadeh [6] presented a heuristic method for solving multi-robot problem. Here, Method is based on the new improved variant of the PSO algorithm, which serves as a global planner. Alternatively, for locale planning and for avoiding obstacles in narrow passages, the Probabilistic Roadmap Method (PRM) is employed. The local and global planners act sequentially until all robots reach their goals. Hao and Xu [7] presented a hybridization algorithm based on Immune Ant Colony Optimization Network (AIN) and Ant Colony Optimization (ACO) for multiple robot path planning. The hybrid AIN-ACO improve the ability of multiple robot system to reach the shortest way. Wang et al. [8] proposed some new methods to solve the problem of multi-robot path planning based on Improved Multi-Objective ABC Algorithm (IMOABC). At first, the foraging mechanism is optimized and some new methods are proposed to calculate the distances of crowding. And also the elimination and restructuring mechanism of food sources. Then, an improved environment map representation method was adopted in which the robot path information is denoted using Cartesian coordinates directly.

In this paper, three case studies based on AMOABC for solving RPP problem are presented the first one for two robots, the second one for three and four robots and the third for five

robots. At each case, the AMOABC algorithm generates optimal paths by the improves the ability for multiple robot system to reach the best 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 Multi-Objective Approaches; Section 5 describes the proposed AMOPSO algorithm and the simulation results and discussion are presented in section 6. Finally, section 7 gives the research conclusions.

2. Problem Formulation

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 [9]:

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 happen in the case of more than one robot trying to take the same position. Also the collision may happen with obstacles 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.

2.1 Two Robots with Shortest Way

The first 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 position (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:

$$f_{short,I}(t) = \sqrt{(X_i(t) - X_f)^2 + (Y_i(t) - Y_f)^2} \quad , i = 1 \ldots N. \qquad (1)$$

The second objective is to find the maximum distance between the paths of the two robot that is needed by each robot from the start position ($X_i(t)$, $Y_i(t)$) to the goal point (X_f, Y_f). The objective function is formulated in the following equation:

$$f_{clear,i}(t) = \frac{1}{\sum_{j=2}^{k} \sqrt{Rx1+Ry1} + \sqrt{Rx2+Ry2}} \qquad (2)$$

where:

$$Rx1 = \left(X_{R1}(t) - X_{R2}(t-1)\right)^2$$
$$Ry1 = \left(Y_{R1}(t) - Y_{R2}(t-1)\right)^2$$
$$Rx2 = \left(X_{R2}(t) - X_{R1}(t-1)\right)^2$$
$$Ry2 = \left(Y_{R2}(t) - Y_{R1}(t-1)\right)^2$$

2.2 Three and Four Robots with the Smoothest and Shortest Way

The first objective function is to find the shortest path for the k-th robot, and the second objective is the smoothest path for the k-th robot, these two objectives can be mathematically expressed in Eq.1 and Eq.3 in which the first equation shows the distance of the robot position to the goal point.

$$f_{smooth,i}(k) = \frac{\cos^{-1}\left[\left(X_i(t-1)-X_f\right)*\left(X_i(t)-X_f\right)+\left(Y_i(t-1)-Y_f\right)*\left(Y_i(t)-Y_f\right)\right]}{\sqrt{\left(X_i(t)-X_f\right)^2+\left(Y_i(t)-Y_f\right)^2}*\sqrt{(X_i(t-1)-X_f)^2+(Y_i(t-1)-Y_f)^2}} \quad (3)$$

2.3 Five Robots with Shortest Way

The objective is to minimize the distance that is needed for each robot from the start position to its goal with minimum time also at each iteration the proposed optimization algorithm take into consideration, the smoothest path. The objective function that is used to minimize the Euclidian distance between the agent current position and the goal point is formulated in Eq.1 and for smoothest path in the Eq.3.

3. Theoretical Background

3.1 Path Planning

The field of robot path planning (RPP) was begun in 1960's. The robot path planning (RPP) problem is very 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 interested in three main matters: accuracy, safety and efficiency. The accuracy and safety issues deal with finding a collision-free path and following the exact addressed path. Efficiency means that the algorithm searches for 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. [10]. Depending on the environment where the robot is located in, RPP can be classified into two types [11]:

1) RPP in static environment: which has fixed obstacles.
2) RPP in dynamic environment: which has both fixed and moving obstacles.

Each of these two types could be further subdivided into a sub-group [11]:

1) Global Path Planning (GPP): A total information about fixed obstacles and a path of moving obstacles is 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).

3.2 Optimization Technique

3.2.1 Standard Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) optimization algorithm is one of the latest population based algorithms, which simulates the behavior of foraging for honey bee colonies. It was invented in 2005 by D. Karaboga [12] with real parameter optimization. With an advantage of employing a fewer control parameter in ABC, the numerical comparisons show that the performance of the ABC algorithm is rival to other population - based algorithms. Because of its flexibility, simplicity and ease of implementation, the ABC algorithm taking a lot of attention and has been used to solve many problems of practical's optimization [12].

In the real bee colony, some missions are finished by specialized individuals. These specialized bees try to make the most of the nectar amount that stored in the hive by using efficient self-organization and division of labour. The minimal model in a honey bee colony of the swarm intelligent forage chosen that the ABC algorithm simulates comprises of three types of bees:

Employed, onlooker and scout bees. half of the colony comprise of employed bees, and the second half contains onlooker. More bees must send to the sources with high quality and should be attracting fewer bees or abandoned the sources with low quality [12].

At first, send the employed bees to promising flower spots by the colony. These bees get and carry flower's nectar to the hive. When they return to the hive, those who have found a spot rated abode a specific quality, put their nectar and go to some patches in the hive named dance floor to share their information with another bee. The communication with other bees is finished with a mysterious dance. If the region that visited by the honeybee is close to the hive, the bee executes a round dance in the hive, and if away, the bee executes a waggle dance. round dance includes information about the quality of nectar for the visited flower spot thus the another bee can find its position by using their smelling sense when they exit from the hive [13].

Waggle dance includes three parts of information about the flower spot: the distance, length from the hive, its direction that can be found and its quality. Onlooker bees keep an eye on the dances in the hive and select the

best flower spots to go to. In fact, flower spots that have higher qualities entice more bees than another that have lower quality. Employed bees that visited zones are abandoned due to low quality have two choices: visit the dance floor and look at dances of another bee to select and then go to a flower spot like an onlooker bee or like a scout bee search around the hive spontaneously for a food source because of some possible external clue or internal motivation [13].

When finding a food source, the bee uses its own ability to memorize the position and then directly starts work to exploit it. After getting the amount of nectar from the food source, bee go back to the hive and unloads an amount of nectar to save it in a food store. Thereafter, the bee has three choices: abandon the dance, food source and then recruit the onlooker bees before going back to the same food source, or go back to the food source without recruiting any onlooker. ABC algorithm is more clarifying in the next subsections [14].

3.2.2 Initialize the population

In the first step, initialize the population of SN individuals by generating it randomly, where *SN* indicates the population size. Every solution X_i *(i = 1,2, · · ·, SN)* representing an individual is a *D*-Dimensional vector. Where, *D* is the number of optimization parameters. And then every solution can be found by using Eq.4 [14].

$$x_{ij} = x_{j\,min} + \left(x_{j\,max} - x_{j\,min}\right).rand(0,1) \quad (4)$$

where $i = 1,2,\cdot\cdot,SN\ and\ j = 1,2,\cdot\cdot,D$. $x_{j\,max}$ and $x_{j\,min}$ are the upper and lower bound of the parameter *j*, respect ively, and rand (0,1) is real number in the range of [0,1]. Then, the fitness for every food source is calculated by

$$fitness = \begin{cases} \frac{1}{1+f_i} & if\ f_i \geq 0 \\ 1 + |f_i| & if\ f_i < 0 \end{cases} \quad (5)$$

where f_i is the cost value of the solution x_i .

3.2.3 Employed bees phase

At this stage, a new food source v_{ij} is generated for the employed bees of food source xi by using the solution search equation

$$v_{ij} = x_{ij} + \emptyset_{ij}.\left(x_{ij} - x_{kj}\right) \quad (6)$$

Where $j \in [1\ldots D]$ and $k \in [1\ldots SN]$ are uniformly distributed random number, k must be different value from i, and $\emptyset_{ij}$ is a uniformly distributed random real number in the range of $[-1, 1]$. After producing a new solution (food source) v_{ij}, it will then be evaluated, then compared with x_{ij} directly. If the fitness of solution v_{ij} is equal or better than of the solution x_{ij}, x_{ij} will be changed with v_{ij} and the individual v_{ij} will become a new member of the

population. Else, food source x_{ij} is kept.

3.2.4 Onlooker bees phase

After that, employed bees finish the search process; on the dance area the employed bees share the nectar information about the food sources and the locations of the food sources with the onlooker bee. Then, every onlooker bee selects a food source with a probability value p_i, which is calculated by using the following equation:

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \quad (7)$$

where $fitness_i$ is the value of fitness for the solution xi, which is proportional to the amount of nectar in the food source.

Obviously, the higher $fitness_i$, the more the probability of choosing the i^{th} food source is. In the onlooker bees' phase, an artificial onlooker bee selects its food source based on the probability value p_i. After choosing probabilistically food source for an onlooker bee, the new neighborhood source is determined by using Eq. (6) and its fitness value is calculated. As was the case in the employed bees' phase, the greedy selection is used among two food sources.

3.2.5 Scout bees phase

If a solution xi that represent a food source is not improved through a predetermined number of trials, named limit, then abandoned the corresponding food source by its employed bee and convert the employed bee which connected with the food source to a scout bee. After that, the scout bee starting the random search for a new solution by using the following equation:

$$x_{ij} = x_{j\,min} + \left(x_{j\,max} - x_{j\,min}\right).rand(0,1) \quad (8)$$

ABC algorithm can be summarized in pseudo code as shown below:

Step1: Initialize the population of solutions x_ij $(i = 1,2,\ldots.SN, j = 1,2,\ldots D)$ using equation of Initialize the population and $cycle = 1$.

Step2: Repeat

Step3: Produce new solutions v_{ij} for the employed bees by using employed bee's phase equation and evaluate them and find fitness using fitness equation for both solutions then apply the greedy selection process.

Step4: Calculate the probability for fitness values p_i (roulette wheel) for the solutions x_{ij} by using probability equation

Step5: Produce the new solutions v_{ij} for the onlookers from

the solutions x_{ij} depending on p_i and then apply the greed selection process.

Step6: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution x_{ij} by using employed bees phase Thereafter, memorize the best solution achieved so far.

Step7: $cycle = cycle + 1$, ***until*** $cycle = MCN$.

4. Multi-Objective Approaches

When the optimization problem contains more than one objective function, the mission of finding one optimal solution or more which is known as multiple objective optimization. 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. As an example, an energy consumption dose not desired at the expense of shortest path along the path. The common methods that are used to deal with multiple objectives optimization are: weighted sum and Pareto front [15].

4.1 Weighted Sum Approach

The weighted sum method combines all multiple objective functions into one scalar, composite objective function using the weighted sum Eq.9 [15].

$$f(x)=\sum_{m=1}^{M} W_m\, fm(x). \quad (9)$$

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

$\sum_{m=1}^{M} W_m = 1$, $W_m \in [0,1]$.

4.2 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 objectives. In the context of Multi-Objective Optimization (MOO), formally, a decision vector $\vec{u} \in \Omega$ i s said to Pareto dominate vector $\vec{v} \in \Omega$, in a minimization context , if and only if:

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

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

In the context of multi-objective optimization, 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 $\vec{\mathrm{F}}(\vec{v})$ [15].

5. Proposed AMOABC Algorithms

In ABC, finding a neighboring food source is defined by Eq. (6) In real bee colony, each employed bee performs a special dance called waggle dance, which consists of three pieces of information about the flower patch: its distance from the hive, its quality and the direction in which it can be found [13]. Therefore, this method is utilized for guiding the bees either to left or right from the current food source (for x-axis) and up or down from the current food source (for y-axis) and keep this direction as long as the fitness increased. While the coefficient $\emptyset_{ij}$ in Eq. (6) is a uniformly distributed random number in the range of $[-1, 1]$, in this paper, parameter $\emptyset_{ij}$ should be adaptive in three cases. Theses adaptive can be defined as follows:

$$\emptyset_{ij} = (b\text{-}a) * Random\ (0,1) + a \quad (10)$$

Where a = - 0.5 and b = 0.5 By try and error, in this case the new $\emptyset_{ij}$ is a uniformly distributed random number in the range of $[-0.5, 0.5]$. The second adaptation rule is:

$$\emptyset_{ij} = \emptyset_{ij} * \frac{iteration}{maximum\ iteration} \quad (11)$$

At first $\emptyset_{\mathrm{ij}}$ = random between [-1,1] and then at each iteration $\emptyset_{\mathrm{ij}}$ is updated according to Eq. (11). In this case the new $\emptyset_{ij}$ is a uniformly distributed random number in the range of $[-1, 1]$. The third adaptation rule is:

$$G = \exp(-\sum_{i=1}^{n} fitness(i)) \quad (12)$$

$$\emptyset_{ij} = 2*G-1 \quad (13)$$

In this adaptive at each iteration $\emptyset_{ij}$ is updated according to fitness of the same iteration, here *n* represent fitness number, and fitness can be obtained according to Eq. (5). So, in the third adaptive rule $\emptyset_{ij}$ located in the range of [-1,1]. In this paper three AMOABC cases are simulated, these cases are listed in Table 1.

Table 1. AMOABC cases

Name	∅ Equation
AMOABC1	10
AMOABC2	11
AMOABC3	12 & 13

6. Simulation Results and Discussion

The proposed algorithms are applied on different known environments with static obstacles. Three case studies are shown below to examine the ability of the AMOABC algorithm in finding the optimal paths. Different tests have been done with the proposed AMOABC algorithms. In the first test, the map dimensions are (11×11) unit distance, and in the second test, the map dimensions are (20×20) unit distance, and in the third test, the map dimensions are (10×10) unit distance, the coordinates for the starting point and for the target point can be seen in Table 2, Table 3, Table 4, Table 5 respectively. The obstacles can be located at any place on the map except at the starting point and at the destination point. The simulations were done with a different number of obstacles using MATLAB R2014a package and executing on the system with 2.60GHz CPU and 2.0G RAM.

The parameters' values of the proposed algorithms and the obstacles' positions utilized in the simulation arenas are explained in the following subsections.

6.1Simulation Parameter Settings

The following parameters of the AMOABC path planning algorithm have been used in the experiment: For all cases, the maximum cycle= 120, limit= 50, and 1 scout bee is generated each time. For case study 1, population size is 6, where 3 are employed bees and the other 3 are onlooker bees; food source NS is 3. For case study 2 and 3 population size is 8, where 4 are employed bees and the other 4 are onlooker bees; food source NS is 4.

b. Start and Target Position Settings for Case Studies

In case study 1, Two robot with different start and target position in the first arena and with different start and same target position in the second arena. The position for each arena are listed in Table 2.

Table 2. Start and target definition for case study 1

Robot No.	Start	Target
1	(0,2)	(4,10)
2	(2.5,4)	(10,10)

Case study 2 is complicated and represented by single arena with three and four robots. Each robot has

different start and target positions. These positions are listed in Table 3 and Table 4.

Table 3. Start and target definition for case study 2/three robot.

Robot No.	Start	Target
1	(10.5,4.5)	(4.5,15.5)
2	(14.5,13.25)	(7,1)
3	(13.25,1.75)	(9,9.5)

Table 4. Start and target definition for case study 2/four robot.

Robot No.	Start	Target
1	(4.5,16)	(17,2.5)
2	(10,0.75)	(15,10.25)
3	(1,2)	(10.75,12)
4	(10.75,14)	(10.75,4.5)

Lastly, case study 3 is more complex and represented by single arena and five robots. Each robot has different start and target positions. These positions are listed in Table 5.

. **Table 5. Start and target definition for case study 5.**

Robot No.	Start	Target
1	(0,0)	(10,10)
2	(2.5,0)	(2.5,10)
3	(7.5,0)	(5,10)
4	(0,2.5)	(7.5,10)
5	(0,7.5)	(10,0)

c. Obstacles' Position Settings for Case Studies

In case study 2, environment with 2 static obstacles situated in different locations. All obstacles' positions (boundary points) are listed in Table 6.

Table 6. Start and target definition for case study 5.

Obstacle No.	1	2
P1	(1,1.25)	(3.5,3.75)
P2	(1.75,1.25)	(4.75,3.75)
P3	(1.75,1.75)	(4.75,4.75)
P4	(1,1.75)	(3.5,4.75)

Case study 2, complex arena with four large obstacles. All obstacles' positions (boundary points) are listed in Table 7.

Table 7. Start and target definition for case study 5.

Obs. No.	1	2	3	4
Point1	(4,2)	(12,4.5)	(2,8)	(4,10)
Point2	(16,2)	(18,4.5)	(14,8)	(10,10)
Point3	(16,4)	(18,18)	(14,18)	(10,18)
Point4	(10,4)	(16,18)	(12,18)	(4,18)
Point5	(10,6)	(16,6)	(12,9)	(6,16)
Point6	(4,6)	(12,6)	(2,9)	(4,14)

Lastly, case study 3 with single complex arena and five irregular obstacles. All obstacles' positions (boundary points) are listed in Table 8.

Table 8. Start and target definition for case study 5.

Obs. No.	Point 1	Point 2	Point 3	Point 4
1	(1.2,2.4)	(2.8,2.3)	(2.5,4)	(1.1,4.5)
2	(5.3,1.9)	(6.1,2)	(6.4,3.3)	(5.5,4.2)
3	(7,5)	(8, 5.2)	(7.5,6.5)	(6.6,6.6)
4	(3.3,6.3)	(4.7,6.1)	(5,7.6)	(3.5,7.7)
5	(1.3, 6.6)	(2.7, 6.9)	(2.2, 7.7)	(1.3, 7.7)

d. Simulation Results for AMOABC Algorithms

For case 2, Table 9 contain the best achieved solutions after 10 runs for AMOABC1, AMOABC 2 and AMOABC 3 with minimum distance (distance from the start to target position) and maximum distance between robot.

The first comparison, it will be between the average of total distance from the start to target point and the maximum distance between the first and second robot for AMOABC1, AMOABC2 and AMOABC3 by using Pareto and weighted sum method.

The best average with shortest distance and maximum distance between the first and second robot is

achieved by Pareto method. The average for total distance for is equal to 9.0832 for the first robot and 9.7890 for the second robot and the average of maximum distance between the two robot is equal to 25.9587. While for weighted sum method the average of total distance for first robot is 9.0374 and 11.0477 for the second robot and the average of clearance is equal to 24.9463.
The second comparison, between the AMOABC and the Hybrid PSO-GSA [5]. According to the results that achieved in **Fig. 1** to **Fig. 4** the AMOABC1, AMOABC2 and AMOABC3 has a maximum clearance and minimum length to reach the target than Hybrid PSO-GSA.

Case 2, for three robots, the best achieved solutions after 10 runs for AMOABC1, AMOABC 2 and AMOABC3 is shown in Table 10 with minimum distance (distance from the start to target position) and smoothness. While Table 11 shows the comparison results in time between Pareto and weighted for AMOABC.
The first comparison, it will be between the average of total time that is needed by Pareto and the weighted sum to find the minimum distance from the start to target point and the smoothness (magnitude). According to the results Pareto has needed 0.0021 second as average of total time while weighted sum needed 0.0025 second. So, here Pareto is faster than weighted sum in 1.19.
The second comparison, between the AMOABC and the Improved PSO [6]. According to the results that achieved in Table 10 and **Fig. 5** to **Fig. 8** the AMOABC1, AMOABC2 and AMOABC3 has faster time to reach the target than Improved PSO. (the dashed line represent the best path achieved by weighted sum while the straight line represents Pareto best path)
For four robots, by comparing the results obtained in the **Figs. 9 – 11** with **Fig 12**, it can be clearly say that the proposed algorithms (AMOABC1, AMOABC2 and AMOABC3) has better ability than IPSO to reach the target with shortest distance to target, minimum time and smoothness.

Table 12 contain the best achieved solutions after 10 runs for AMOABC1, AMOABC2 and AMOABC3 with minimum distance (distance from the start to target position), smoothness and time. While Table 13 shows the comparison results in time between Pareto and weighted sum for AMOABC.

The first comparison, it will be between the average of total time that is needed by Pareto and the weighted sum to find the minimum distance from the start to target point and the

smoothness (magnitude). According to the results Pareto has needed 0.0021 second as average of total time while weighted sum needed 0.0041 second. So, in this case Pareto is 1.96 time faster than weighted sum.

The second comparison, between the AMOABC and the Improved PSO [6]. According to the results that achieved in Table 12 and **Fig. 9** to **Fig. 12** the AMOABC1, AMOABC2 and AMOABC3 has faster time to reach the target than Improved PSO. (the dashed line represent the best path achieved by weighted sum while the straight line represents Pareto best path)

Finally, for case3, Best time achieved by Immune Ant Colony Optimization Network Algorithm [7], is listed in Table 15. By comparing the results achieved in Table 15 the AMOABC1, AMOABC 2 and AMOABC 3 has a minimum time to reach the target than Immune Ant Colony Optimization Network Algorithm and Pareto reach the target in 4.22 times less than weighted sum and 723.9 times less than Immune Ant Colony Optimization Network Algorithm.

7. Conclusion

In this paper, the results of a detailed investigation of the Adaptive Multi Objective Artificial Bee Colony (AMOABC) algorithms applied to multi robot path planning optimization problem were presented. The effectiveness of these algorithms on the RPP was tested based on different environments and different simulation parameters; the results achieved were compared with some previous works. From the collected results and the comparison, the following can concluded:

1. AMOABC1, AMOABC2 and AMOABC3 has a better ability to get away from local optimums with a quickest convergence than the MOABC.
2. Pareto method give a better and fast result in compared with weighted sum method. Where, the best Pareto path is better than best weighted sum path with multi- objective.
3. In case study 1, the proposed algorithms generate optimal path for two robots with respect to two objectives: shortest path to target and maximum distance between robot's path. By comparing the results achieved, it can be clearly say that the proposed algorithms AMOABC has a better ability than GSA-PSO to reach the target with shortest distance to target and maximum distance between robot's path (clearance).
4. In case study 2 and 3, the proposed algorithms trying to

achieve the best path with three functions: find the shortest path, smoothness and minimum time. According to the results achieved by proposed algorithms in case of five robots, AMOABC has a minimum time to reach the target than Immune Ant Colony Optimization Network Algorithm (AIN-ACO) and Pareto reach the target in 4.22 times less than weighted sum and 723.9 times less than AIN-ACO Algorithm.

Table 9. Simulation results for case 1.

	Robot No.	Pareto Method		Weighted Sum Method	
		Total Dist.	Max. Dist.	Total Dist.	Max. Dist.
AMOAB1	1	9.0902	24.796	9.0379	24.601
	2	9.7857		9.8049	
AMOAB2	1	9.0785	27.256	9.0342	24.593
	2	9.7898		13.603	
AMOAB3	1	9.0809	25.824	9.0401	25.645
	2	9.7916		9.7351	

Table 10. Simulation results for case 2/three robot.

Table 11. Comparison of the average runtimes of the AMOABC with their standard after 600 run for three robot case.

	Robot No.	Pareto Method			Weighted Sum Method		
		Total Dist.	Smoothness	Time	Total Dist.	Smoothness	Time
AMOAB1	1	17.986	138.8049	0.0016319	18.499	180.6482	0.0022833
	2	23.033	41.8090		23.26	42.4443	
	3	24.094	50.9040		24.042	45.4985	
AMOABC2	1	17.973	137.4910	0.0022422	20.023	155.4062	0.002549
	2	24.523	36.1194		23.252	43.5437	
	3	24.277	58.3056		24.204	58.4204	
AMOABC3	1	17.99	125.0118	0.0023847	17.996	125.5722	0.002619
	2	23.154	41.2277		23.066	42.6099	
	3	24.057	68.5620		24.06	67.4265	

Algorithms	Average	Standard Deviations
AMOABC1 – Pareto	9.2018 * e-06	8.636 * e-06
AMOABC2 – Pareto	7.9331 * e-06	2.0473 * e-06
AMOABC3 – Pareto	4.2219 * e-06	5.2966 * e-06
AMOABC1 – Weighted Sum	0.0036	0.0113
AMOABC2 – Weighted Sum	0.0037	0.0127
AMOABC3 – Weighted Sum	0.0041	0.0131

Table 12. Simulation results for case 2/ four robot.

Robot No.	Pareto Method			Weighted Sum Method		
	Total Dist.	Smoothness	Time	Total Dist.	Smoothness	Time
1	25.268	80.6051	0.0016975	25.247	80.8084	0.0039721
2	22.057	72.0332		22.088	71.3188	
3	17.565	71.0066		17.572	70.7204	

4	21.507	72.7805		21.443	72.7805	
1	25.652	80.2377	0.0026271	25.313	81.9754	0.0042993
2	22.739	68.9661		19.507	73.4348	
3	17.709	67.9659		17.591	74.5802	
4	21.1	69.2769		21.646	69.2769	
1	25.473	79.4289	0.0019177	25.529	82.2786	0.0040767
2	19.292	81.2843		20.217	74.3082	
3	17.668	80.7694		17.678	92.7093	
4	21.193	69.7633		21.214	69.7633	

Table 13. Comparison of the average runtimes of the AMOABC with their standard after 600 run for four robot case.

Algorithms	Average	Standard Deviations
AMOABC1 – Pareto	7.3838 * e-05	8.0384 * e-05
AMOABC2 – Pareto	9.4423 * e-05	1.7411 * e-05
AMOABC3 – Pareto	8.8262 * e-05	3.5512 * e-05
AMOABC1 – Weighted Sum	0.0086	0.0285
AMOABC2 – Weighted Sum	0.0095	0.0322
AMOABC3 – Weighted Sum	0.0078	0.0287

Table 14. Simulation results for case 3.

Robot No.	Total Dist.	Smoothness
1	14.17	0.0027
2	10.041	0.0052
3	10.308	0.01
4	10.889	0.0070
5	12.571	0.0052

Table 15. Time for case 3.

Algorithms	Time	
	Pareto Method	Weighted Sum Method
AMOABC 1	0.00095295	0.0043194
AMOABC 2	0.00080491	0.003554
AMOABC 3	0.00094624	0.0034577
AIN-ACO	0.6525	

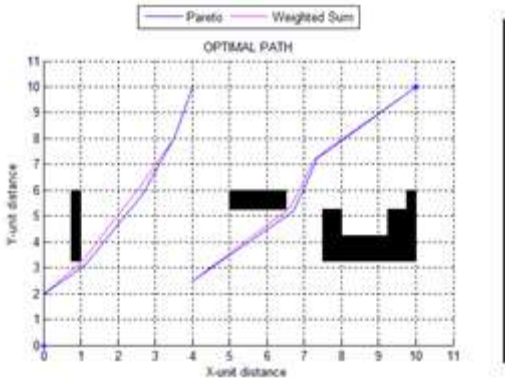

Fig. 1. Case 1 result achieved by AMOABC1

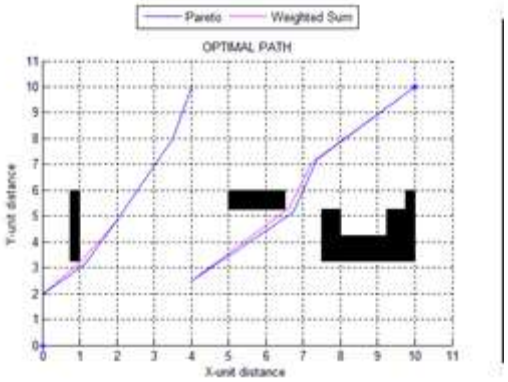

Fig. 3. Case 1 result achieved by AMOABC3

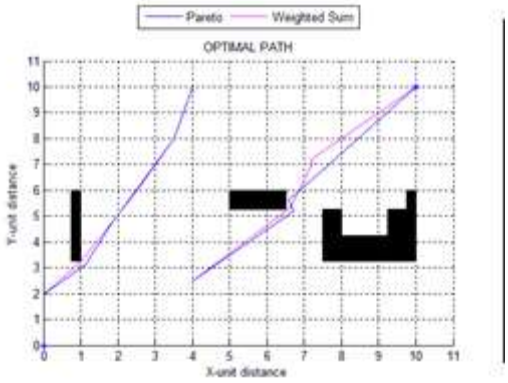

Fig. 2. Case 1 result achieved by AMOABC2

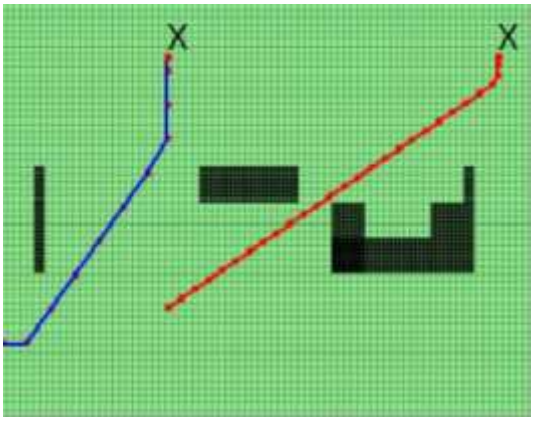

Fig. 4. Case 1 result achieved by PSO-GSA [6]

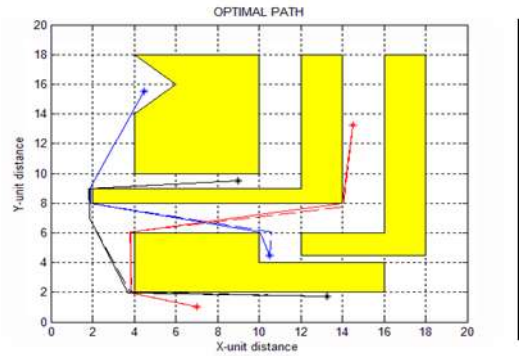

Fig. 5. Case 2 result achieved by AMOABC1

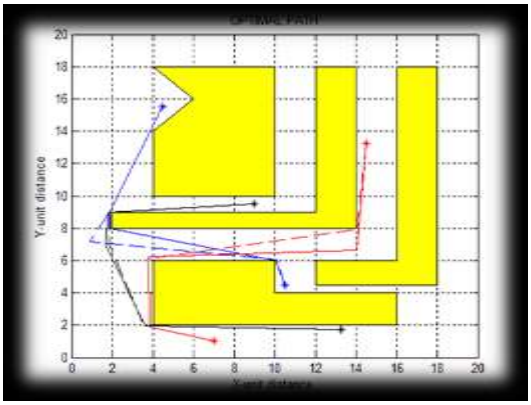

Fig. 6. Case 2 result achieved by AMOABC2

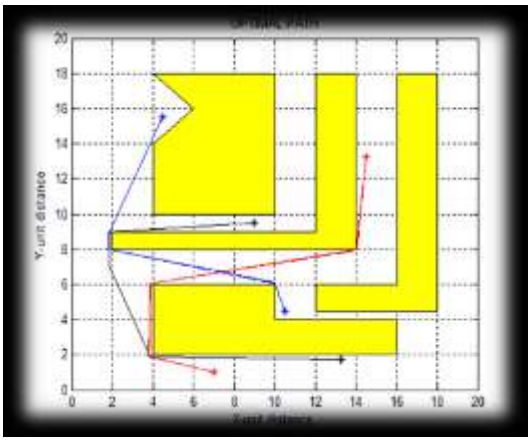

Fig. 7. Case 2 result achieved by AMOABC3

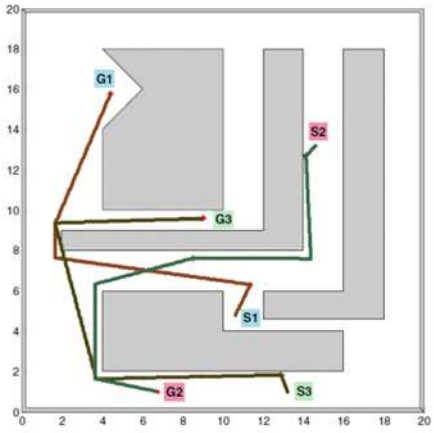

Fig. 8. Case 2 results achieved by IPSO [6]

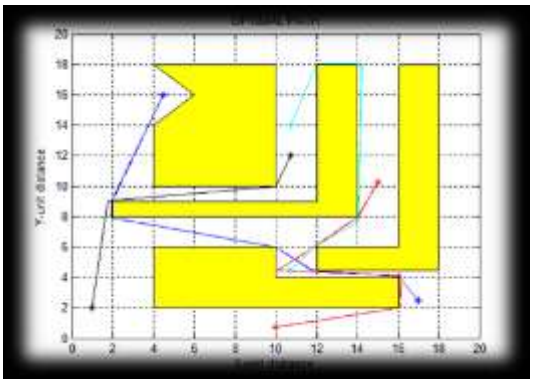

Fig. 9. Case 2 result achieved by AMOABC1

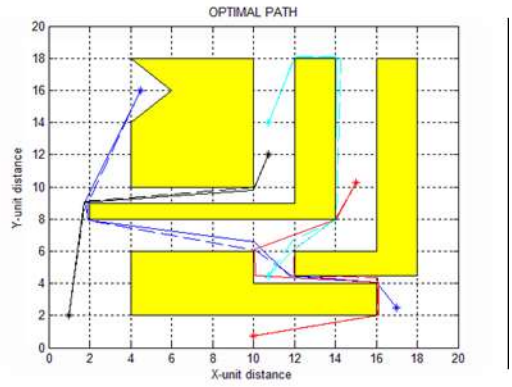

Fig. 10. Case 2 result achieved by AMOABC2

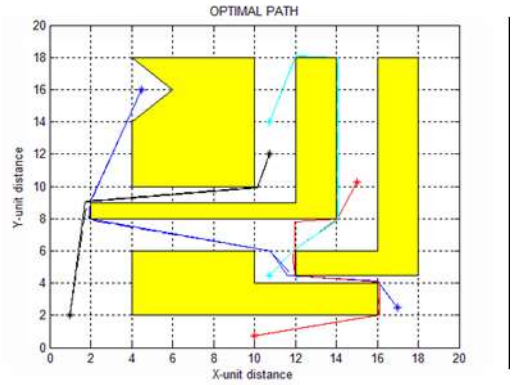

Fig. 11. Case 2 results achieved by AMOABC3

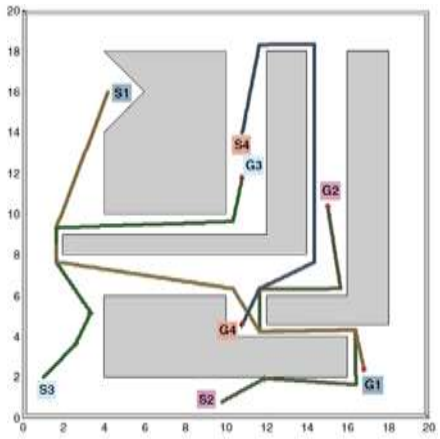

Fig. 12. Case 2 results achieved by IPSO [6]

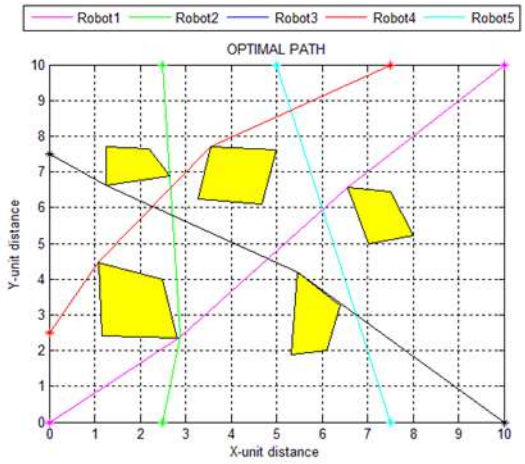

Fig. 13. Case 3 results achieved by AMOABC

7. References

1- H. Li, Yang, S. X. Simon and Y. Biletskiy, "Neural Network Based Path Planning for a Multi-Robot System with Moving Obstacles," In proceeding of Automation Science and Engineering, pp. 163-168, 2008.

2- B. Wu and C. Qian, " Differential Artificial Bee Colony Algorithm for Global Numerical Optimization," Journal of Computer, Vol. 6, No. 5, pp. 841-848, May 2011.

3- P. Bhattacharjee, P. Rakshit, I. Goswami and A. Konar, "Multi-robot path-planning using artificial bee colony optimization algorithm", Third World Congress on Nature and Biologically Inspired Computing (NaBIC), Salmanca, Spain, pp. 219–224, 19-21 oct. 2011.

4- M. Agarwal and P. Goel, "Path Planning of Mobile Robots using Bee Colony Algorithm," MIT International Journal of Computer Science & Information Technology, Vol. 3, No. 2, pp. 86–89, 2013.

5- C. Purcaru, R.-E. Precup, D. Iercan, L. -O. Fedorovici, M.E. Petriu and E.-I. Voisan, "Multi-

Robot GSA- and PSO-Based Optimal Path Planning in Static Environments", In Proceedings of 9th Workshop on Robot Motion and Control (RoMoCo), Kuslin, Poland, pp. 197-202, 3-5 July 2013.

6- E. Masehian and D. Sedighizadeh, "An Improved Particle Swarm Optimization Method for Motion Planning of Multiple Robots", Springer Tracts in Advanced Robotics, Vol. 83, pp. 175-188, 2013.

7- W. Hao and X. Xu, "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, 27-29 June 2014.

8- Z. Wang, M. Li, L. Dou, Y. Li, Q. Zhao and J. Li, "A Novel Multi-Objective Artificial Bee Colony Algorithm for Multi-Robot Path Planning", IEEE International Conference on Information and Automation, Lijiang, China, pp.481 - 486, 8-10 Aug. 2015.

9- J. A. Abdulsaheb and N. H. Abbas, "An Adaptive Multi-Objective Particle Swarm Optimization Algorithm for Multi-Robot Path Planning", Baghdad University, Journal of Engineering, Baghdad, Iraq, Vol. 22, No. 7, pp. 164-181, 2016.

10- K. M. Han, "Collision Free Path Planning Algorithms for Robot Navigation Problem," Master Thesis, University of Missouri, Columbia, August 2007.

11- H. Miao, "Robot Path Planning in Dynamic Environments using Simulated Annealing Based Approach," Master thesis, Queensland University of Technology, Queensland, Australia, March 2009.

12- D. Karaboga, "An Idea based on Honey Bee Swarm for Numerical Optimization," Technical Report, Erciyes University, Engineering Faculty, Computer Engineering Department, pp. 1-10, 2005.

13- M. H. Saffari and M. J. Mahjoob, "Bee Colony Algorithm for Real-Time Optimal Path Planning of Mobile Robots, " 5th International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (ICSCCW), Famagusta, Cyprus, pp. 1-4, 2-4 Sept. 2009.

14- W. Xiang, S. Ma and M. An, " yAn Improved Artificial Bee Colony Algorithm with Multiple Search Operators," Journal of

Computational Information Systems, Vol. 9, No. 8, pp. 3130-3139, August 2013.

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

تطوير خوارزمية مستعمرة النحل الاصطناعية متعددة الوظائف لتخطيط المسار لاكثر من روبوت

استاذ مساعد نزار هادي عباس

جعفر احمد عبدالصاحب

قسم الهندسة الكهربائية

كلية الهندسة – جامعة بغداد / العراق

الخلاصة

في هذا البحث تم عرض افضل مسار عن طريق استخدام خوارزمية مستعمرة النحل الاصطناعية متعددة الوظائف المعدلة لدراسة ثلاث حالات. في الحالة الاولى، اثنين من الروبوتات يحاولان الوصول الى نهايات مختلفة عن طريق اقصر مسار ممكن من نقطة البداية الى نقطة النهاية واكبر مسافة ممكنة بين مسار الروبوت الاول ومسار الروبوت الثاني. في الحالة الثانية تحاول الخوارزمية الحصول على المسار الامثل لأقصر مسار ممكن واكثر المسارات مرونة لثلاث واربع من الروبوتات. اخيراً ، خمس من الروبوتات يحاولون الوصول الى اهداف مختلفة مع تجنب التصادم فيما بينهم ومحاولة الحصول على اقصر مسار ممكن واكثر المسارات مرونة. النتائج اظهرت ان الخوارزمية المطورة تمتلك قدرة افضل للخروج من الأمثلية 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 المفتاحية: **نظام متعدد الروبوت، تخطيط المسار، طرق الوظائف المتعددة ، خوارزمية مستعمرة النحل متعددة الوظائف.**