An ABC-Optimized Reciprocal Velocity Obstacles Algorithm for Navigation of Multiple Mobile Robots †

Ziyad T. Allawi[1] and Turki Y. Abdalla[2]

[1] College of Education for Humanitarian Studies, University of Baghdad, Baghdad, Iraq
[2] Department of Computer Engineering, College of Engineering, University of Basrah, Basrah, Iraq

E-mail: ziyad.allawi@gmail.com, protryounis@yahoo.com

Received: 21/5/2014

Accepted: 19/5/2015

Abstract - In this paper, a new optimization method for the Reciprocal Velocity Obstacles (RVO) is proposed. It uses the Artificial Bee Colony Optimization (ABC) for navigation control of multiple mobile robots with kinematic constraints. RVO is used for collision avoidance between the robots, while ABC is used to choose the best path for the robot maneuver to avoid colliding with other robots and to get to its goal faster. This method is applied on 24 mobile robots facing each other. Simulation results have shown that this method outperformed the ordinary RVO when the path was arbitrarily chosen.

Keywords- Artificial Bee Colony, Multiple Mobile Robots, Navigation, Reciprocal Velocity Obstacles

† *This paper has been presented in ECCCM-2 Conference* and accredited for publication according to IJCCCE rules.

1. Introduction

The robotic technologies have been widely employed in many applications. Nowadays, robot systems have been applied in factory automation, entertainment, space, etc... Recently, more and more researchers takes interest in the robot which can help people in our daily life, such as service robot, office robot, security robot, home robot, and so on.

Other than control and coordination of multiple mobile robots, one of the central problems in this area is motion planning among multiple moving mobile robots. Each robot navigates independently without explicit communication with the other mobile robots. Therefore, the basic problem as navigating a single agent to its goal location without colliding with the obstacles and the other mobile robots in the environment can be formulated [1].

The problem of computing a collision-free path for a robot moving among dynamic obstacles is not only of interest to robotics but also has been widely studied for crowd simulation in computer graphics, virtual environments, video gaming, traffic engineering and architecture design, where each agent can be considered as a virtual human, a moving car, or an individual pedestrian [1]. It is important in many robotics applications, including automated transportation systems, automated factories, and applications involving robot human interactions, such as robotic wheelchairs [2].

The problem of collision avoidance between the robots is harder than the moving obstacles because the robots are not simply moving without considering their environment; they are also intelligent decision-making entities that try to avoid collisions as well. Simply considering them as moving obstacles may lead to *oscillations* if the other entity considers all other robots as moving obstacles as well. Therefore, the reactive nature of the other entities must be specifically taken into account in order to guarantee that collisions are avoided [3]. An alternative to complete planning is to plan for the robot as it acts, taking new sensor inputs as they arrive and planning locally. Some of the prominent work in this area is based on the Velocity Obstacles (VO) [1].

In this paper, the problem of real-time navigation for multi-robot motion planning in dynamic environments is addressed. It introduces an optimized navigation method that combines the *reciprocal velocity obstacles* (RVO) [1] collision avoidance navigation method with the *artificial bee colony* optimization algorithm (ABC) [4]. RVO constructs the velocity obstacle regions for a given robot induced by other robots and chooses feasible velocity and orientation intervals considering the kinematic constraints of the robot. ABC inspects these intervals and chooses the best velocity and orientation that ensures a collision-free path for the robot which makes it closer to the desired goal. This method is simulated on 24 mobile robots facing each other; each robot tries to reach a goal that is behind its opposite partner. This method is compared with the same RVO method with arbitrary chosen configurations and the results were obtained for different simulation parameters.

2. Related Works

Many recent works have considered the problem of navigating a robot in an environment composed of dynamic obstacles and other moving robots. Some of the simplest approaches predict where the dynamic obstacles may be in the future by extrapolating their current velocities, and let the robot avoid collisions accordingly. However, such techniques are not sufficient when a robot encounters other robots, because treating the other robots as dynamic obstacles

overlooks the reciprocity between robots. In other words, the other robots are not passive but are actively trying to avoid collisions. Therefore, the future trajectories of other robots cannot be estimated by simply extrapolating their current velocities, since this would inherently cause undesirable oscillations in their trajectories [5].

Numerous motion planning algorithms have been developed for mobile robots in static environments. In [6], the notion of a car-like robot was formalized, and the fact that a path for a holonomic robot lying fully in open regions of the configuration space can always be transformed into a feasible path for a nonholonomic robot was proven. Laumond *et al.* [6] also provided an algorithm to generate a feasible path for a nonholonomic robot from a path found for a holonomic robot. Approaches applicable to mobile robots have been developed for complete trajectory planning among moving obstacles as in [7] and [8].

Several variations of the velocity obstacle formulation have been proposed for multi-robot systems, generally by attempting to incorporate the reactive behavior of the other entities in the environment. Variations such as RVO [1], [9], recursive probabilistic velocity obstacles [10], [11], and common velocity obstacles [12] use various means to handle reciprocity, but each has their own shortcomings. Specifically, the approach of [11] may fail to converge, while other concepts [1], [12] are limited to dealing with only two robots.

Other work has focused mainly on follow-the-leader behavior when navigating robots in real-world settings [13], [14]. Also, there is a large body of work on centrally coordinating the motions of multiple robots [15]. However, there is little work on navigation of multiple independent robots to arbitrary goals in real world settings while taking into account the reactive behavior of other robots.

Integration of the ABC algorithm with mobile robotics is still in the early stages. A recent research proposed by Bhattacharjee *et al.* [16] used ABC for optimizing multi-robot path planning. The proposed method tries to determine a trajectory of motion to known targets where ABC is used to minimize the path travelled by the robots including obstacle and collision avoidance.

3. Research Methodology

3.1. Reciprocal Velocity Obstacles:

It is a navigation method used for robot motion planning in dynamic environments. It consists of selecting avoidance maneuvers to avoid static and moving obstacles in the velocity space, based on the current positions and velocities of the robot and obstacles. It is a first-order method since it does not integrate velocities to yield positions as functions of time [17].

The avoidance maneuvers are generated by selecting robot velocities outside of the VO, which represent the set of robot velocities that would result in a collision with a given obstacle that moves at a given velocity at some future time. To ensure that the avoidance maneuver is dynamically feasible, the set of avoidance velocities are intersected with the set of admissible velocities, defined by the robot's kinematics constraints.

Fiorini and Shiller [17] were the first who introduced this approach to be used in the path planning and navigation of mobile robots in dynamic environments which comprise the presence of static and moving obstacles (or other moving robots). They applied the approach upon the intelligent vehicles negotiating highway traffic.

The key features of the VO as mentioned in [17] were:

1. VO provides a simple geometric representation of potential avoidance maneuvers;
2. Any numbers of moving obstacles can be avoided by considering the union of their VO's;
3. It unifies the avoidance of moving as well as stationary obstacles; and
4. It allows for the simple consideration of robot dynamics.

Since the VO had been introduced, some developments were carried out upon the original algorithm. Jur Berg contributed most of these developments. Reciprocal VO was introduced by Berg *et al.* [1], generalized VO was proposed by Wilkie *et al.* [18] and hybrid reciprocal VO was proposed by Snape *et al.* [19], [20]. All these approaches were applied on the navigation of multiple mobile robots and it was mainly used for inter-robot collision avoidance. For instance, a hybrid reciprocal VO for collision-free and oscillation-free navigation of multiple mobile robots was presented in [20]. Each robot senses its surroundings and acts independently without central coordination or communication with other robots. The approach used both the current position and the velocity of other robots to compute their future trajectories in order to avoid collisions. The approach was reciprocal and avoids oscillations by explicitly taking into account that the other robots sense their surroundings as well and change their trajectories accordingly.

The method presented in this paper simultaneously determines actions for many robots that each may have different objectives. The actions are computed for each robot independently without communication among the robots. The method uses ABC algorithm to obtain the optimum collision-free velocity which guarantees the collision-free motion for each robot in the environment.

The fundamental configuration of the original VO is shown in the figure below [18]:

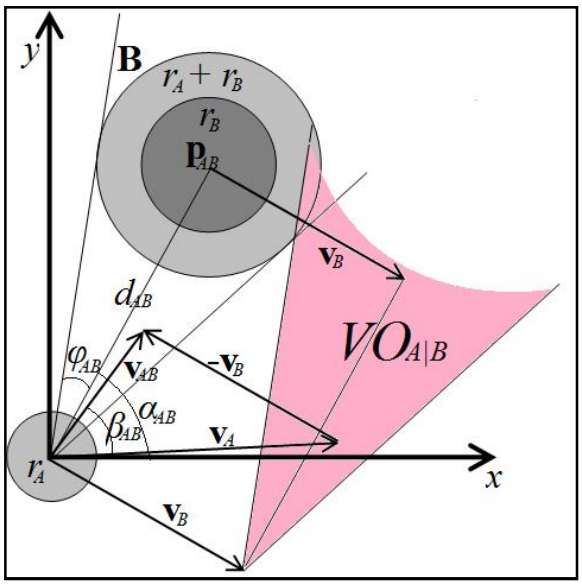

Figure 1. The Velocity Obstacle of Robot A induced by Robot B

In Figure 1, for a disc-shaped robot A (light gray) of radius r_A, located at $\mathbf{p}_A$, has a velocity of $\mathbf{v}_A$ and a disc-shaped robot B (dark gray) of radius r_B, located at $\mathbf{p}_B$, has a velocity of $\mathbf{v}_B$, the velocity obstacle for A induced by B, denoted $VO_{A|B}$ (the pink-colored cone), is the set of all velocities for A that would, at some point in the future, result a collision with B. This set is defined geometrically. Let the robot A be a single point in the origin and $\mathbf{B}$ be a disc (light gray ring encircling dark gray disc) centered at $\mathbf{p}_{AB}$ with a radius equal to the sum of A's and B's ($r_A + r_B$). If B is static (i.e. not moving), a cone of velocities for A could be defined that would lead to a collision with B as the set of rays shot from the origin that intersect the boundary of $\mathbf{B}$. To derive a velocity obstacle from this, the cone is translated by the velocity $\mathbf{v}_B$ of B, as shown in figure. Briefly, the general representation of the Standard Velocity Obstacles appears below [2]:

$$VO_{A|B} = \{\mathbf{v} \mid \exists t > 0 :: \mathbf{p}_A + t(\mathbf{v} - \mathbf{v}_B) \in \mathbf{B}\} \quad (1)$$

RVO originates from the mutual behavior of the two robots to avoid oscillation which may happen due to the change of direction. In this method, instead of choosing a new velocity for each robot that is outside the other robot's VO, a new velocity is chosen that is the average of its current velocity and a velocity that lies outside the other robot's VO.

The general representation of RVO is shown below [1]:

$$RVO_{A|B} = \{\mathbf{v} \mid \exists t > 0 :: \mathbf{p}_A + t(\mathbf{v} - (1-\alpha) * \mathbf{v}_A - \alpha * \mathbf{v}_B) \in \mathbf{B}\} \quad (2)$$

where α is a constant which refers to the effort share between the robots. In the standard RVO, $\alpha = 0.5$.

This method performs simpler calculations than the original *RVO* in [1] using the shape of RVO cone (base angle and axis) to check if $\mathbf{v}_A$ is inside the cone or outside it; and permits the use of optimization algorithms.

The algorithm begins by calculating the distance between the two robots d_{AB} and its argument α_{AB} with respect to the Cartesian coordinates as in:

$$\begin{aligned} \mathbf{p}_{AB} &= \mathbf{p}_B - \mathbf{p}_A = [x, y] \\ d_{AB} &= \|\mathbf{p}_{AB}\| = \sqrt{x^2 + y^2} \\ \alpha_{AB} &= \text{atan2}(y, x) \end{aligned} \quad (3)$$

The line d_{AB} and its argument α_{AB} represent the cone axis of symmetry. Then, the cone base semi-angle φ_{AB} can be calculated as shown below:

$$\varphi_{AB} = \sin^{-1}\left(\frac{r_A + r_B}{d_{AB}}\right) \quad (4)$$

The parameters $\mathbf{v}_B$, d_{AB}, α_{AB}, and φ_{AB} can be used to check the robot A's possibility of collision with robot B as follows:

It is assumed $\mathbf{v'}_A$ to be an arbitrarily chosen velocity of the robot A, subjected to the nonholonomic constraints of that robot. First, the magnitude of velocity difference between the two robots and its argument with respect to the Cartesian coordinates, v_{AB} and β_{AB}, respectively are calculated as shown below:

$$\begin{aligned} \mathbf{v}_{AB} &= (\mathbf{v'}_A - \tfrac{\mathbf{v}_A + \mathbf{v}_B}{2}) = [x, y] \\ v_{AB} &= \|\mathbf{v}_{AB}\| = \sqrt{x^2 + y^2} \\ \beta_{AB} &= \text{atan2}(y, x) \end{aligned} \quad (5)$$

The sufficient condition of being $\mathbf{v'}_A$ inside the $RVO_{A|B}$ is when ψ_{AB}, the absolute difference between α_{AB} and β_{AB}, is smaller than φ_{AB} as in:

$$\begin{aligned} &\psi_{AB} = |\beta_{AB} - \alpha_{AB}| \\ &\begin{cases} \mathbf{v'}_A \in RVO_{A|B} & \text{if } \psi_{AB} \le \varphi_{AB} \\ \mathbf{v'}_A \notin RVO_{A|B} & \text{if } \psi_{AB} > \varphi_{AB} \end{cases} \end{aligned} \quad (6)$$

If $\mathbf{v'}_A$ lies in the $RVO_{A|B}$, then changing the magnitude and/or the direction of $\mathbf{v'}_A$ is mandatory for escaping collision. Choosing $\mathbf{v}_A^*$ (optimum velocity) for escaping from all the robots in the environment may be carried out by using heuristic methods or optimization techniques considering subjection to the nonholonomic constraints of the robot A in linear and angular velocities.

One can calculate the minimum possible time of collision (if there is one). This time is calculated only if $\mathbf{v'}_A$ lies in the $RVO_{A|B}$, because otherwise the time will be infinite as in below:

$$t_c = \frac{d_{AB}\cos\psi_{AB} - \sqrt{(r_A + r_B)^2 - d_{AB}^2 \sin^2\psi_{AB}}}{v_{AB}} \quad (7)$$

The term under the square root will be positive only if $\psi_{AB} \le \varphi_{AB}$.

The time of collision t_c should be compared with the sampling time of the simulation τ. Although $\mathbf{v}_A$ lies in the $VO_{A|B}$, but the collision will be certain only when τ is greater than t_c. If the

opposite happen (i.e. τ is smaller than t_c), the collision will not happen in the next time sample although the collision is still possible, then a penalty formula can be used to choose the best velocity among a set of candidate velocities. The penalty formula used in this work is shown below [1]:

$$p(\mathbf{v}'_A) = \frac{k}{t_c} + \left\| \mathbf{v}^G_A - \mathbf{v}'_A \right\| \qquad (8)$$

where $\mathbf{v}^G{}_A$ is the goal-directed velocity of the robot *A,* and *k* is a constant. The penalty is increased when the robot velocity diverges from the goal velocity and the collision time is short. The optimum velocity $\mathbf{v}_A^*$ will be the velocity which has the least penalty (i.e. close to the goal velocity and out from the *RVO*).

This algorithm may be used for optimization concerns where the optimization algorithm should use these previous assumptions to select another velocity arbitrarily until an optimum velocity $\mathbf{v}_A^*$ is found which guarantees a collision-free path for the robot *A* in the overall environment.

The RVO pseudo algorithm is shown below:

Function V = *VelocityObstacles*(*n*)
// *n* = number of robots in the environment
for *i* = 1 **to** *n* **do**
$\mathbf{p}_i$ = position of robot *i* [x_i, y_i, θ_i]; // θ_i may be directed toward a target
$\mathbf{v}_i = v_{i\max}$*[cosθ_i, sinθ_i];
r_i = radius of the robot;
end for
for *i* = 1 **to** *n* **do**
for *j* = 1 **to** *n*, $j \neq i$ **do**
d_{ij} & α_{ij} = the *RVO* cone axis and its argument as in Equation 3;
φ_{ij} = the *RVO* cone semi-angle as in Equation 4;
end for
loop
f = 0;
for *j* = 1 **to** *n*, $j \neq i$ **do**
v_{ij} & β_{ij} = the magnitude and argument of velocity difference as in Equation 5;
Use Equation 6 to check if $\mathbf{v}_i$ lies in the $RVO_{A|B}$; if so, *flag* = 1; else *flag* = 0;
t_c = collision time (if $\mathbf{v}_i$ lies in the $RVO_{A|B}$) as in Equation 7;
Check if $t_c > \tau$; if so, *flag* = 0;
f = *f* + *flag*;
end for
Use heuristics or an optimization method to find $\mathbf{v}_i^*$ that minimizes *f* considering the nonholonomic constraints of the robot *i* or by using Equation 8;
until $\mathbf{v}_i^*$ is found;
$\mathbf{V}_i = \mathbf{v}_i^*$;
end for
return V;
end.

3.2. Artificial Bee Colony:

ABC algorithm is one of the modern global optimization algorithms. It is a branch of the bio-inspired optimization algorithms which include Genetic Algorithms (GA), Ant Colony Algorithms (ACO), Differential Evolution (DE) and Particle Swarm Optimization (PSO) algorithms. It is as simple as ACO, PSO and DE algorithms, and uses only common control parameters such as colony size and maximum cycle number. ABC as an optimization tool provides a population-based search procedure in which individuals called foods positions are modified by the artificial bees with time and the bee's aim is to discover the places of food sources with high nectar amount and finally the one with the highest nectar. In ABC system, artificial bees fly around in a multidimensional search space and some (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates, and adjust their positions. Some (scouts) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, attempting to balance exploration and exploitation process [3].

This optimization algorithm was introduced by Karaboğa [21] in 2005. He presented ABC as a branch of the *Swarm*

Intelligence. He proposed the main steps of the algorithm and tested it on 3 benchmark functions, and then he continued the development of ABC algorithm and published two papers with Başturk [2], [22]. They compared ABC performance with GA, DE and PSO on a group of benchmark numerical multi-dimensional functions. They found that ABC outperformed all other mentioned algorithms.

Several modifications and improvements were carried out upon ABC since then. Most of these modifications were carried out for sake of avoiding local optimum results which usually occur in the multi-dimensional systems. For instance, Karaboğa and Akay [4] modified ABC to be executable for constrained optimization problems. In [5], Karaboğa; again, modified ABC by introducing a new parameter to control the frequency of perturbations on the predicted solutions. Quan and Shi [23] introduced an improved artificial colony algorithm. They presented a new search iteration operator based on the fixed point theorem of Contractive Mapping. Narasimhan [24] presented the Parallel ABC. He divided the optimization process on several processors rather than one processor and let the solutions on a local memory and made them available for all the bees for further improvements. Liu and Cai [25] made some modifications on the original algorithm by adding randomized distribution, bit hyper-mutation and a novel crossover operator to improve the performance of the original algorithm. Bi and Wang [26] improved the algorithm by introducing fast mutation in which the scout bee behavior was replaced by opposite-based learning strategy.

In a real bee colony, some tasks are performed by specialized individuals. These specialized bees try to maximize the nectar amount stored in the hive using efficient division of labor and self-organization. The minimal model of swarm-intelligent forage selection in a honey bee colony which the ABC algorithm simulates consists of three kinds of bees: employed bees, onlooker bees and scout bees. Half of the colony consists of employed bees, and the other half includes onlooker bees. Employed bees are responsible for exploiting the nectar sources explored before and giving information to the waiting bees (onlooker bees) in the hive about the quality of the food source sites which they are exploiting. Onlooker bees wait in the hive and decide on a food source to exploit based on the information shared by the employed bees.

Every bee colony has scouts that are the colony's explorers. The explorers do not have any guidance while looking for food. They are primarily concerned with finding any kind of food source depending on an internal motivation or based on possible external clues. As a result of such behavior, the scouts are characterized by low search costs and a low average in food source quality. Occasionally, the scouts can accidentally discover rich, entirely unknown food sources. In the case of artificial bees, the artificial scouts could have the fast discovery of the group of feasible solutions as a task [24].

In the ABC algorithm, the position of a food source represents a possible solution to the optimization problem, and the nectar amount of a food source corresponds to the profitability (fitness) of the associated solution. Each food source is exploited by only one employed bee. In other words, the number of employed bees is equal to the number of food sources existing around the hive (number of solutions in the population). The employed bee whose food source has been abandoned becomes a scout. The

complete mathematical procedure of ABC is explained in details in [4].

4. Results & Discussions

The design procedure is done under MATLAB® environment by using 24 identical mobile robots. It will be assumed that the kinematic robot model will be used, where each robot is assumed to have a simple shape (circle) moving in a two-dimensional workspace. Also, each robot has perfect sensing, and is able to infer the exact shape, position and velocity of other robots in the environment. The robots are positioned in a circle perimeter and facing its center as in Figure 2. The goals are located in the same place of their opponents. So, the robot will try to maneuver all other 23 robots to escape and reach its goal. For comparison, the RVO method is used with arbitrary chosen velocities while ABC was inserted into RVO in the second scenario.

The single particle is a 2-dimension vector holding the magnitude and direction of robot's velocity. The penalty constant k and the population size SN are varied. The other parameters are fixed. Their values were MCN = 100, $limit$ = 100 and SP = 100.

The robot specifications are shown in Table 1.

Table 1. Specifications of the Mobile Robot

Specification	Value
Wheel Rotational Velocity	20 rad/s
Angular Steering Velocity	5 rad/s
Radius of the Robot	10 cm
Radius of the Wheel	5 cm
Maximum Heading Velocity	100 cm/s

Table 2 and 3 illustrate the mean robot travelling distances (in cm) for the two methods respectively for various k and SN (Direct Displacement = 1000).

Table 2: Mean Distances for robots in RVO method

RVO	Distance
k = 5	1133
k = 10	1158
k = 20	1207
k = 50	1262

Table 3: Mean Distances for robots in ABC-RVO method

ABC-RVO	SN = 10	SN = 20	SN = 50	SN = 100
k = 5	1133	1101	1121	1100
k = 10	1170	1161	1142	1149
k = 20	1190	1172	1191	1215
k = 50	1229	1266	1232	1237

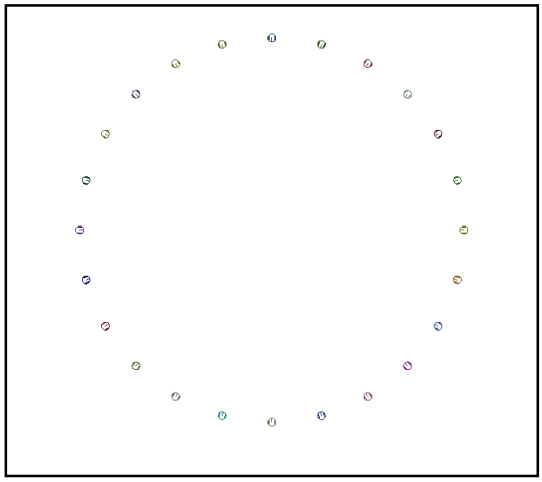

Figure 2. Robot Positions for ABC-RVO algorithm, SN = 100, k = 5

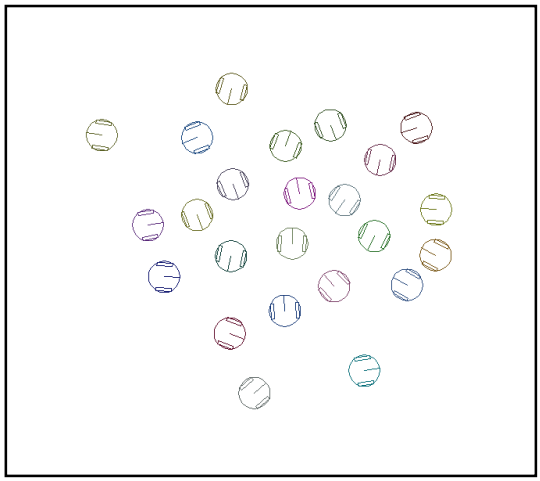

Figure 3. Robots are maneuvering each other

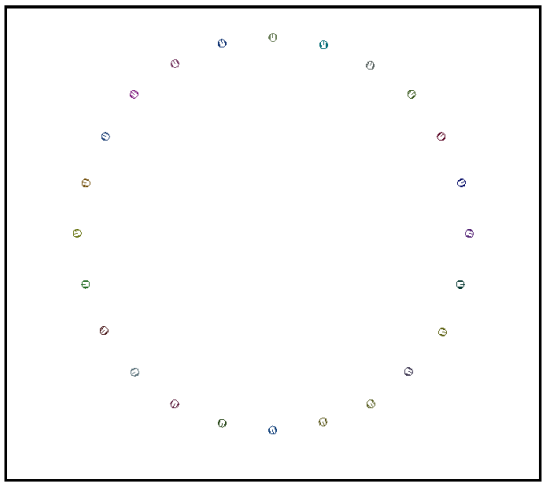

Figure 4. All robots are in the goals.

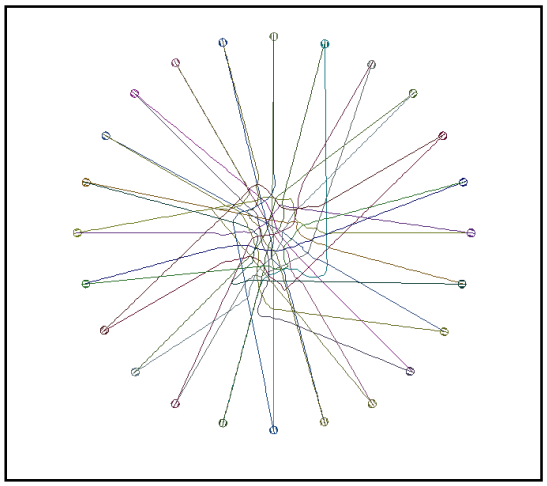

Figure 5. Robot Paths in the above scenario

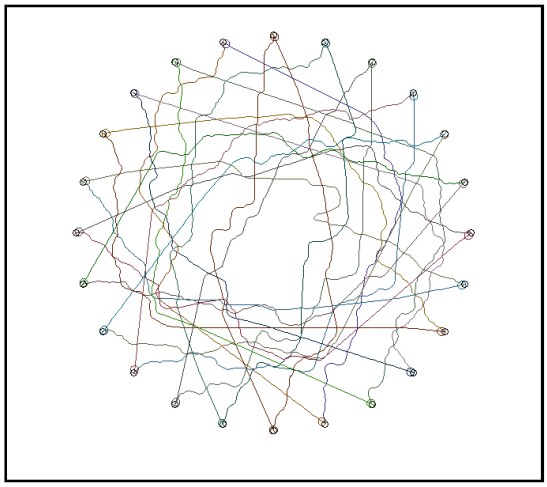

Figure 6. Robot Paths for RVO method with arbitrary velocity choice

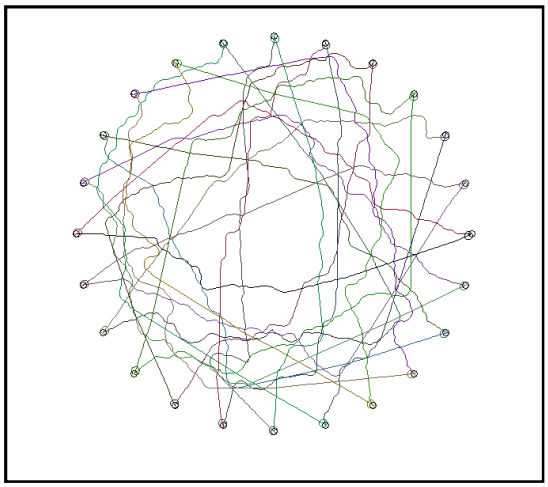

Figure 7. Robot Paths for ABC-RVO method for SN = 20, k = 50

In the above Figures (2 to 5), it is seen the behavior of the robots in a scenario of facing each others with the parameters k = 5 and SN = 100. In Figure 2, the robots are heading directly towards their goals despite of the possible collisions. Afterwards, the robots begin to maneuver when the distances between them are being smaller and smaller. ABC tries to select the best next move for the robots depending on their current configuration and the other robots configuration. The maneuver takes place in a small area of the environment, that's because k is small. It is seen in Figure 3 (zoomed in) that some of the robots are close to each other but they will not collide because that ABC chooses the best collision free path for everyone. Then, the robots succeeded to escape the maneuvering space and moved directly towards their destinations. In Figure 4, all the robots have arrived safely to their goals.

Figure 5 illustrates the paths which all robots take. It is seen that the maneuvering took place in a small area, and the robots moved in a pure line from their origins to the maneuvering area and from that area to the goals after escaping.

It is seen that the robot paths have reasonable oscillations in some regions of the environment, that's because k is small.

Choosing k is very important in that situation; it should be small enough to cancel all oscillations that may be happened in the course of maneuvering. But if one chooses k very small, the process may fail due to collisions before the maneuvering.

Figures 6 and 7 show the environment when choosing high k. It's clear that the robot paths are oscillatory from the beginning and do not be a direct path unless the robot escapes maneuvering. It is seen in the Figure 6 the oscillations are more than the oscillations in Figure 7, that's because of the arbitrary choice of velocities in ordinary RVO method.

If one looks to Tables 2 and 3, it is seen that ABC-RVO outperforms itself and the ordinary RVO in the mean robot distances when k = 5, and becomes more and more oscillatory with the increase in travelling distance when k is increased; therefore, one should choose k wisely to ensure collision-free and oscillation-free navigation of multiple robots.

Increasing SN aids in finding the optimal next move of the robot. It is seen in Table III that the minimum mean distances travelled by the robots occur when SN = 100, which is 1100 cm. As in all optimization algorithms, increasing agent size is important to find the optimal solution but it consumes time; therefore one should choose a moderate value for SN if he wants to apply this algorithm in real-time environments.

5. Conclusions

It can be concluded from this work that using the optimization methods in the navigation of multiple mobile robots helped in some way to increase the efficiency of robot maneuvering. This is clear from the results shown in Tables 2 and 3. ABC-RVO method has succeeded in creating a collision-free, oscillation-free optimum path for all the robots provided that choosing the appropriate values of k and SN.

References

[1]. J. van den Berg, M. Lin and D. Manocha, "*Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation*," IEEE International Conference on Robotics and Automation, 2008, pp. 1928-1935.

[2]. D. Karaboğa and B. Başturk, "*On the performance of artificial bee colony (ABC) algorithm*," Elsevier Journal of Applied Soft Computing, Vol. 8, 2008, pp. 687-698.

[3]. D. Karaboğa and B. Akay, "*A comparative study of Artificial Bee Colony algorithm*," Elsevier Journal of Applied Mathematics and Computation, Vol. 214, 2009, pp. 108-132.

[4]. D. Karaboğa and B. Akay, "*A modified Artificial Bee Colony (ABC) algorithm for constrained optimization problems*," Elsevier Journal of Applied Soft Computing, Vol. 11, 2011, pp. 3021-3031.

[5]. B. Akay and D. Karaboğa, "*A modified Artificial Bee Colony algorithm for real-parameter optimization*," Elsevier Journal of Information Sciences, Vol. 192, 2012, pp. 120-142.

[6]. J. P. Laumond, P. E. Jacobs, M. Taix and R. M. Murray, "*A motion planner for nonholonomic mobile robots*," IEEE Transactions on Robotics and Automation, Vol. 10, No. 5, 1994, pp. 577-593.

[7]. E. Frazzoli, M. Dahleh and E. Feron, "*Real-time motion planning for agile autonomous vehicles*," Journal of Guidance Control and Dynamics, Vol. 25, No. 1, 2002, pp. 116-129.

[8]. D. Hsu, R. Kindel, J-C. Latombe and S. Rock, "*Randomized kinodynamic motion planning with moving obstacles*," The International Journal of Robotics Research, Vol. 21, No. 3, p. 233, 2002.

[9]. S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha and P. Dubey, "*Clear Path: Highly parallel collision avoidance for multi-agent simulation,*" in Proceedings of the ACM SIGGRAPH Eurographics Symposium of Computer and Animation, 2009, pp. 177-187.

[10]. C. Fulgenzi, A. Spalanzani, and C. Laugier., "*Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid*," in Proceedings of the IEEE International Conference of Robotics and Automation, 2007, pp. 1610-1616.

[11]. B. Kluge and E. Prassler, "*Reflective navigation: Individual behaviors and group*

behaviors," in Proceedings of the IEEE International Conference of Robotics and Automation, 2004, pp. 4172–4177.

[12]. Y. Abe and M. Yoshiki, "*Collision avoidance method for multiple autonomous mobile agents by implicit cooperation*," in Proceedings of the IEEE RSJ International. Conference of Intelligent Robotic Systems, 2001, pp. 1207-1212.

[13]. K. C. Ng and M. M. Trivedi, "*A neuro-fuzzy controller for mobile robot navigation and multirobot convoying*," IEEE Transactions of Systems, Man and Cybernetics B, Vol. 28, No. 6, 1998, pp. 829-840.

[14]. S. Carpin and L. E. Parker, "*Cooperative motion coordination amidst dynamic obstacles*," in Proceedings of the International Symposium of Distributive Autonomous Robotic Systems, 2002, pp. 145-154.

[15]. S. M. LaValle, "*Planning Algorithms*," Cambridge, U.K. Cambridge University Press, 2006.

[16]. P. Bhattacharjee, P. Rakshit, I. Goswami (Chakraborty), A. Konar and A. Nagar, "*Multi-Robot Path-Planning Using Artificial Bee Colony Optimization Algorithm*," IEEE 3rd World Congress on Nature and Biologically Inspired Computing, 2011, pp. 219-224.

[17]. P. Fiorini and Z. Shiller, "*Motion planning in dynamic environments using velocity obstacles*," The International Journal of Robotics Research, Vol. 17, No. 7, 1998, pp. 760-772.

[18]. D. Wilkie, J. Berg and D. Manocha, "*Generalized Velocity Obstacles*," The IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 5573-5578.

[19]. J. Snape, J. Berg, S. Guy and D. Manocha, "*Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles*," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009, pp. 5917-5922.

[20]. J. Snape, J. Berg, S. Guy and D. Manocha, "*The Hybrid Reciprocal Velocity Obstacle*," IEEE Transactions on Robotics, Vol. 27, No. 4, 2011, pp. 696-706.

[21]. D. Karaboğa, "*An Idea Based On Honey Bee Swarm for Numerical Optimization*," Technical Report-TR06, Computer Engineering Department, Engineering Faculty, Erciyes University, Kaiseri, Turkey, 2005.

[22]. D. Karaboğa and B. Başturk, "*A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm*," Springer Journal of Global Optimization, No. 39, 2007, pp. 459-471.

[23]. H. Quan and X. Shi, "*On the Analysis of Performance of the Improved Artificial Bee Colony Algorithm*," IEEE 4th International Conference on Natural Computations, 2008, pp. 654-658.

[24]. H. Narasimhan, "*Parallel Artificial Bee Colony (PABC) Algorithm*," IEEE World Congress on Nature-Biology Inspired Computations, 2009, pp. 306-311.

[25]. X. Liu and Z. Cai, "*Artificial bee colony Programming Made Faster*," IEEE 5th International Conference on Natural Computations, 2009, pp. 154-158.

[26]. X. Bi and Y. Wang, "*An Improved Artificial Bee Colony Algorithm*," IEEE 3rd International Conference on Computer Research and Development, 2011, Vol. 2, pp. 174-177.