Swarm and Evolutionary Computation 28 (2016) 14–28

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Regular Paper

A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning

P.K. Das [a,*], H.S. Behera [a], B.K. Panigrahi [b]

[a] Department of Computer Science and Engineering and Information Technology, VSSUT, Burla, Odisha, India
[b] Department of Electrical Engineering, IIT, Delhi, India

ARTICLE INFO

Article history:
Received 22 July 2015
Received in revised form
29 September 2015
Accepted 26 October 2015
Available online 7 January 2016

Keywords:
Multi-robot path planning
Average total trajectory path deviation
Average untraveled trajectory target distance
Average path Length
IPSO–IGSA
Energy optimization

ABSTRACT

This paper proposed a new methodology to determine the optimal trajectory of the path for multi-robot in a clutter environment using hybridization of improved particle swarm optimization (IPSO) with an improved gravitational search algorithm (IGSA). The proposed approach embedded the social essence of IPSO with motion mechanism of IGSA. The proposed hybridization IPSO–IGSA maintain the efficient balance between exploration and exploitation because of adopting co-evolutionary techniques to update the IGSA acceleration and particle positions with IPSO velocity simultaneously. The objective of the algorithm is to minimize the maximum path length that corresponds to minimize the arrival time of all robots to their respective destination in the environment. The robot on the team make independent decisions, coordinate, and cooperate with each other to determine the next positions from their current position in the world map using proposed hybrid IPSO–IGSA. Finally the analytical and experimental results of the multi-robot path planning were compared to those obtained by IPSO–IGSA, IPSO, IGSA in a similar environment. The Simulation and the Khepera environment result show outperforms of IPSO–IGSA as compared with IPSO and IGSA with respect to optimize the path length from predefine initial position to designation position ,energy optimization in the terms of number of turn and arrival time.

1. Introduction

The Path planning problem in mobile robotics is considered as a complex task. It has studied from the paper [8] that work determines a path for the robot to reach in predefined goal location from a specified starting location without hitting with various obstacles in the given Environment. The path planning problem has been classified into different categories. One of the classifications is static and dynamic path planning based on the environmental information. In the static path planning, the obstacles and goals are motionless. But, in the dynamic path planning the obstacles and goals are moving in the environment in each time and also the environment is changing in every time. Another classification is local and global path planning. Robot navigates through the obstacles by steps and determines its next position to reach at the goal by satisfying constraints like path, time and energy-optimality [1–7] with the help of the local path planning scheme. In global planning, the robot decides the entire collision free path before its movement towards the goal from a specified initial position. The above mentioned global planning is termed as *offline planning* [32]. Local path-planning, which includes navigation and online planning, sometimes referred to as navigation only in the literature. The phrase motion planning that includes the notion of time with the position of a robot on a planned trajectory, is often used in the context of path-planning. Motion planning thus takes care of planning the path with some resource management or constraints over time. In the last decades, Significant progress has been made on a single robot and multi-robot in motion planning [11,13,15] by using of some traditional and heuristic approaches such as potential field method [14], visibility graph-based, Voronoi-diagram [31], real time A* algorithm [8,9,19], Simulated Annealing and neural network [10] and evolutionary [16,17] algorithms. But, in the classical approach needs more time complexity in large problem space and trapping in local optimum are drawbacks. Therefore, solving these problems using classical approach is impractical. Hence heuristic algorithms have become more popular to solve optimization problem. Heuristic algorithms maintain a good balance between diversification and intensification to achieve both efficient global and local search. Therefore, authors have consented on solving multi-robot optimization problem using heuristic algorithms. In a multi-robot path planning problem, each robot has a specified initial and goal position in a given environment and each robot have to plan their

* Corresponding author.
E-mail addresses: daspradipta78@gmail.com (P.K. Das), hsbehera_india@yahoo.com (H.S. Behera), bkpanigrahi@ee.iitd.ac.in (B.K. Panigrahi).

http://dx.doi.org/10.1016/j.swevo.2015.10.011
2210-6502/

collision free path without hitting any of the colleagues or obstacles present in the map through offline or online approach. The obstacles present in the environment may be static or dynamic. However, in this paper we have considered static obstacles in the given environment for the robots and robot is treated as dynamic obstacles for other robots. The path planning problem for multi-robot can be solved by two different approaches such as centralized or distributed approach. The cost or objective function and the constraints for computing the path for all the robots are considered together in the centralized approach [16,17]. Whereas, in the distributed planning [22], each robot determined its collision free trajectory path towards the goal independently without making collision with static obstacles or colleagues. The multi-robot navigational problem has divided into two smaller problems such as velocity planning and path planning. In the first phase, each robot constructs the individual path by satisfying the optimum path for each robot. In the velocity planning, each robot avoids the collision with obstacles and the teammates. In our study, we have proposed a novel meta-heuristic optimization approach to carry out the multi-robot navigational problem. Different meta-heuristic optimization algorithms have been used to generate the optimum trajectory collision free for each robot.

Particle swarm optimization (PSO) with mutation operator [49] has been used to develop an algorithm for path planning for a mobile robot. Multi-objective optimization problem has formulated for obstacle avoidance in a dynamic environment and solved it by PSO [50]. An algorithm for robot path planning has developed using PSO of Ferguson Splines [51]. A smooth path planning of a mobile robot has solved using stochastic PSO [52]. In [21] multi-objective PSO-and NPSO based algorithms for robot path planning. In [57] PSO is used in an obstacle avoidance scenario in which robots are meant to navigate between static and dynamic obstacles. In [53–56] Area Extension PSO (AEPSO) and Cooperative AEPSO (CAEPSO) are employed as navigators of a swarm of robots in dynamic and static environments. AEPSO and CAEPSO take advantage from macro scoping modelling of PSO in addition to extra heuristics that utilize concepts such as reinforcement learning and cooperative learning. Multi-objective particle swarm optimization (MPSO) has been used to solve the path planning problem under uncertainty to minimize the path length and risk degree [20]. In [43] used two multi-objective path planning models to find a safe path by minimizing the energy consumption. A stochastic path planning scheme has been proposed [44] for a mobile robot to find safety and smooth path. Multi-objective PSO with self-adaptive mutation operation has been proposed [45] to solve the path planning problem in an environment with obstacles and danger source. A PSO used endocrine regulation mechanism [46] to update the particle behaviors by the interaction of neural and endocrine systems and reduced the local convergence with help of momentum factor has been utilized to plan a robot path. Binary PSO [47] has been proposed to find the global optimal path and a path is encoded with some vertices of the polygon type's obstacle present in the environment. Similarly, a PSO has embedded with potential field's approach [48] to update particles and avoid the obstacles with the help of potential field method. Gravitational search algorithm (GSA) and on a particle swarm optimization (PSO) algorithm applied to multiple mobile robot on holonomic wheeled platforms [58]. Dijkstra algorithm and bat algorithm [68] has been proposed to find the global optimal path. Cuckoo search (CS) algorithm has been applied for mobile robot path planning in an unknown or partially known environment with variety of static obstacles [69]. The different intelligent algorithm had also been used to solve robot path planning such as memetic algorithm [63,70], culture algorithms [64], biogeography particle swarm optimization algorithm (BPSO) [65]. Glowworm Swarm Optimisation (GSO) [66] has been used to solve multiple source localisation tasks through real robot experiments and GSO [67] has also been used for pursuing multiple mobile targets using single and two source cases. Firefly algorithm (FA) [71] has been proposed for solving the path planning problem by improving the solution quality and convergence speed. Multi-robot path planning problem is well known for optimisation problem, recently, so many swarm intelligence algorithms have been proposed to solve multi-constrained problem such as grey wolf optimizer (GWO) [72], chicken swarm optimisation (CSO) [73], krill herd (KH) [74], monarch butterfly algorithm (MBA) [75]. These algorithms are inspired by swarm behaviour of grey wolves, chicken, krill, and butterfly, respectively. Gravitational search algorithm (GSA) based approach has been applied for generating an optimal path for a robot travelling in partially unknown environments in the presence of multiple (static or dynamic) obstacles [30]. Path planning of Uninhabited Aerial Vehicle has been solved using improved GSA [59]. Differential Evolution (DE) [17] has used for multi-robot navigation in a static environment and performance of the algorithm has used for minimizing the path length. Hybridization of meta-heuristic algorithms such as ACO-GA [32], PSO–GSA [33], and Hybrid Evolutionary Algorithm Based on Tree Structure Encoding [34] has been solved for multi-robot path planning. However, we have hybridized improved PSO (IPSO) and IGSA for better performance with respect to IPSO [24,27] and IGSA.

However, PSO suffers from premature convergence in the evolutionary process while dealing with complex problems such as some real world navigation based optimization problem like the solution of path planning in multi-robot. PSO also depends on users to tune control parameters such as inertia weight, social and cognitive coefficients and velocity clamping in order to achieve the required solution. For instance, as iteration increase the initial weight in PSO is mostly decreasing linearly from 0.9 to 0.4 in order to emphasize on the exploitation. However, there is no mechanism for significant hasty movements in the search space for PSO and this makes the poor performance of PSO [61]. Therefore, the structure of PSO algorithm needs further improvement for achieving an optimal solution to the real world problems. GSA is one of the meta-heuristic algorithms, which is based on the law of gravity and mass interaction, and implements law of motion and Newtonian gravity. The advantages of GSA are (1) easy to implement with higher computational efficiency; (2) few parameters to adjust, but the disadvantages of this algorithm as follow (1) if premature convergence occurs, there will not be any recovery for this algorithm; (2) the algorithm loses its ability to explore and then becomes inactive only after becoming convergence [62]. Due to the above difficulties in GSA, further improvements are required for the optimal solution to the complex problem. The main idea of hybridizing of improved PSO and GSA to integrate the ability of exploitation in PSO and exploration in GSA to produce both algorithms' strength [60]. Therefore, authors motivated to hybrid the improved PSO and GSA for maintaining a good balance between diversification and intensification to achieve global optima.

The above proposed work has been focused on the minimizing the path length of the mobile robot and not considered the path deviation and energy consumption as a key factor. In this work, we have focused on the minimizing the path cost, path deviation and energy consumption for each robot with the help of the proposed algorithm. This paper contributes to establish a novel optimization algorithm by a hybrid of IPSO and IGSA. The main idea of this hybridization is to embed the social and cognitive behavior of IPSO with the Newtonian gravity concept of IGSA by co-evolutionary techniques. Then, this novel hybrid improved particle swarm optimization and improved gravitational search algorithm (IPSO–IGSA) is proposed to solve the multi-robot path planning in a cluster environment, where some of the obstacles are static and

some of the obstacles are dynamic in nature. This proposed approach effectively improved the exploration and exploitation in IPSO–IGAS because of simultaneously updating the particle positions with IPSO velocity and IGSA acceleration.

The main objective of this paper may be summarized as follows: (i) we study the problem of multi-robot path planning in a clutter environment and formulated the above problem as multi-objective optimization problem with constraints; (ii) novel method to the solution of optimal trajectory path generation for multi-robot path planning problem using IPSO–IGSA is proposed in this article; (iii) the proposed algorithm has been applied for multi-robot path planning in a clutter and dynamic environment and obtained results are compared to other optimization algorithms like IPSO, IGSA; (iv) the performance of the proposed IPSO–IGAS, as an optimizing tool in solving multi-robot path planning problem, is applied in the simulation as well as Khepera-II environment and result are presented; (v) the performance matrix of the proposed approach is successfully validated in simulation and Khepera-II.

For realizing multi-robot path planning problem with hybridization of IPSO–IGSA [25], a fitness function is constructed for the IPSO to compute the next positions for each robot to produce the optimal trajectory path leading towards their respective goals by updating the particle position with IPSO velocity and IGSA acceleration. The fitness function of the IPSO represents two components. First, selection of the next position of the robot in the optimal path of the objective function and the second one is representing the constraint to avoid collision with teammates and static obstacles. In this paper, we enhance our implementation of the hybrid IPSO–IGSA technique, to compute the optimal trajectory path of all the robots from specified initial positions to fixed goal positions in the environment and with an objective to minimize the path distance for each robot. The result shows that the algorithm can improve the solution quality in a reasonable amount of time and a successful optimal path is designed with the help of hybrid IPSO–IGSA technique by avoiding the dynamic as well as static obstacles in its path towards the goal.

The remaining part of the paper is outlined as follows. The classical particle swarm optimization and improved particle swarm optimization are described briefly in the Section 2. The proposed improved gravitational search algorithm is presented in the Section 3. Theoretical description and its algorithm of the hybrid IPSO–IGSA for path planning of multi-robot is presented in Section 4. Formulation of the problem for multi-robot path planning is elaborated in Section 5. Implementation of multi-robot path planning problem using hybrid IPSO–IGSA Algorithm is described in detail in Section 6. The simulation result of the implementation is presented in Section 7. In Section 8 experiment is conducted in Khepera-II environment and finally, conclusions of the work in the paper are presented in Section 9.

2. Particle swarm optimization (PSO)

2.1. Classical PSO (CPSO)

The CPSO is a stochastic population based, bio-inspired evolutionary optimization algorithm, which was originally introduced by Kennedy and Eberhart (1995), which utilizes swarm intelligence to achieve the goal of optimization. It is based on intelligent collective behavior of schools fish or bird flocks. In the classical PSO algorithm, each member of the population is known as a particle in a D-dimensional search and set of particles is called a swarm. The velocity parameter of the CPSO is dynamically updated by the particles own experience and flying experience of its accompaniment. The members of the entire population share the information among individual to change each particle position to find the best position in the search space. The advantage of the CPSO over other optimization algorithm is easy to implement and few parameters are to be adjusted.

Let N be the population size. In each generation k, the velocity and position of the particles are updated using Eq. (1).

$$V_i^d(t+1) = V_i^d(t) + C_1.\phi_1.\left(x_{pbest_i}^d - x_i^d(t)\right) + C_2.\phi_2.\left(x_{gbest}^d - x_i^d(t)\right)$$
$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1) \quad (1)$$

where $x_i = (x_{i1}, x_{i2}, \ldots\ldots x_{iD})$ represent the current position vector of the particle $i(1 \le i \le N)$ in a D-dimensional search space, $V_i = (V_{i1}, V_{i2}, \ldots\ldots V_{iD})$ represent the velocity of the ith particle, $C_1(C_1 \neq 0)$ and $C_2(C_2 \neq 0)$ are the acceleration constants, ϕ_1 and ϕ_2 are two random numbers in the range [0, 1]. x_{pbest} is the previous best position of the ith particle in generation k, x_{gbest} is the previous global best position among all the particles in generation k . If $C_1 = 0$, then PSO algorithm is converted to social-only model. Similarly, if $C_2 = 0$, then it becomes a cognition-only model.

2.2. Improved particle swarm optimization

To bring a balance between the exploration and exploitation characteristics of PSO, Shi and Eberhated proposed a PSO on inertia weight in which velocity of each particle is updated and claimed that a larger the value of the inertia weight will be provided a global search, while smaller the value will be provided local search. So, it needs to change the inertia weight dynamically to adjust the search capability dynamically. Therefore, there are several proposals to modify the PSO algorithm by changing the inertia weight value in an adaptive manner in each iteration. In this paper, we have improved PSO (IPSO) in the terms of adaptive weight adjustment and acceleration coefficients to increase the convergence rate to optimum value in PSO, the classical PSO equation modified according to the following form:

$$V_i^d(t+1) = w_i V_i^d(t) + C_1.\phi_1.(x_{pbest_i}^d - x_i^d(t)) + C_2.\phi_2.(x_{gbest}^d - x_i^d(t))$$
$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1) \quad (2)$$

The local best value in IPSO can be computed as

$$pbest_i(t+1) = \begin{cases} pbest_i(t), & \text{if } O_{bj}(x_i(t+1)) \succ O_{bj}(pbest_i(t)) \\ x_i(t+1), & \text{if } O_{bj}(x_i(t+1)) \prec O_{bj}(pbest_i(t)) \end{cases} \quad (3)$$

where O_{bj} stands for the fitness function of the moving particles and the global best position is obtained as

$$gbest(t) = \min\{O_{bj}(pbest_1(t)), O_{bj}(pbest_2(t)), \ldots\ldots O_{bj}(pbest_N(t))\} \quad (4)$$

The convergence rate of the PSO has been improved by fine tuning of its parameter with the help of several techniques. These techniques usually change the PSO update equations without altering the inherent structure of the algorithm. The velocity during the previous time step is scale it by a scale factor inertia weight (w) to update a new velocity every time as the particle moves the search space. The large value of the inertia weight provides the global search where as the small value of the inertia weight makes a local search. The search ability is adjusted dynamically by dynamic change of inertia weight. Here, the adaptive change of inertial weight proposed in [12] is used and presented in the Eq. (5). Empirical experiments have been performed in the past with an inertia weight varies in [0.9, 0.4].

$$w_i = w_{min} + (w_{max} - w_{min})\frac{dist_i}{max\_dist} \quad (5)$$

where $dist_i$ is the current Euclidean distance of particle i from global best is defined in Eq. (6) and initial value and final value of the inertia weight is 0.4 and 0.9 respectively and max_dist is the

maximum distance of a particle from global best in that generation as defined in Eq. (7).

$$dist_i = \left(\sum_{d=1}^{D} gbest_d - x_i^d \right)^{1/2} \tag{6}$$

$$max\_dist = \arg \max_i (dist_i) \tag{7}$$

Similarly, the fixed value set for the acceleration coefficients (conventionally fixed to 2.0). With the large value of the social component C_2 in comparison with a cognitive component C_1 leads particles to a local optimum prematurely and relatively high value of cognitive components results to wander the particles around the search space. The quality of the solution quality is improved by modifying cognitive and social coefficient term in such a way that the cognitive component is reduced and social component is increased as generation proceeds. The modification of the coefficients are made (for tth generation) using the following Eqs. (8) and (9).

$$C_1 = C_{1i} - \left(\frac{C_{1i} - C_{1f}}{Max\_Iter} \right) t \tag{8}$$

$$C_2 = C_{2i} + \left(\frac{C_{2f} - C_{2i}}{Max\_Iter} \right) t \tag{9}$$

where C_{1i}, C_{1f}, C_{2i} and C_{2f} are initial and final values of cognitive and social component acceleration factors respectively and *Max_Iter* is the maximum number of allowable iterations.

The accelerating and inertial weights are the major factor in the numerical results [12,23] to improve the accuracy. The algorithm ability has improved by combining PSO with other search techniques [26]. To escape from local minima and increase the diversity of population, PSO combine with some evolutionary operators [28,29,35] such as crossover, mutation and selection. Hence, to improve the performance of PSO, other evolutionary algorithm such as GSA is used in this paper.

3. Improved gravitational search algorithm (GSA)

Recently the scientific community has gained the interest on GSA. It is a meta-heuristic optimization algorithm inspired by nature which is based on the Newton's law of gravity and the law of motion [39]. GSA is grouped under the population based approach and is reported to be more natural. The algorithm is planned to improve the performance in the exploration and manipulation capabilities of a population based algorithm, based on gravity rules.

GSA is based on the two important formulas about Newton Gravity Laws given by Eqs. (10) and (11). Eq. (10) is the gravitational force equation between the two particles, which is directly proportional to their masses and inversely proportional to the square of the distance between them. But in GSA instead of the square of the distance, only the distance is used. Eq. 11 is the equation of acceleration of a particle when a force is applied to it [39].

$$F = G\frac{M_1 M_2}{R^2} \tag{10}$$

$$a = \frac{F}{M} \tag{11}$$

G is the gravitational constant, M_1 and M_2 are masses and R is the distance between two masses, F is the gravitational force, and a is the acceleration. Based on these formulas, the heavier object with more gravity force attracts the other objects as it is seen in Fig. 1.

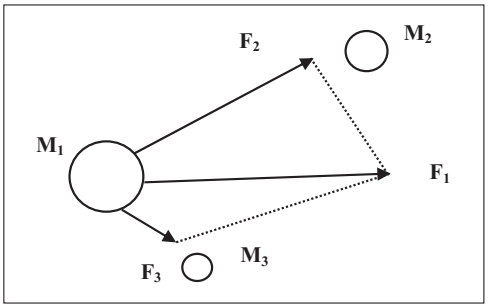

Fig. 1. The Newton gravitational force representation.

In GSA, each mass (agent) has four characteristics, namely position, inertial mass, active gravitational mass, and passive gravitational mass. The position of the mass corresponds to a solution of the problem, and the fitness function are used to determine the gravitational and inertial masses [40,41].

3.1. Agents initialization

Consider a system with N masses in which position of the ith mass is defined as follows:

$$X_i = \left(x_i^1, \ldots, x_i^d, \ldots, x_i^n \right) \text{ for } i = 1, 2, \ldots, N \tag{12}$$

where x_i^d is the position of ith mass in dth dimension and n is dimension of the search space.

3.2. Fitness and best fitness computation

$worst(t)$ and $best(t)$ are defined as follows for minimization case:

$$worst(t) = \max_{i \in p} fit_i(t), \quad p = 1, 2, \ldots, N \tag{13}$$

$$best(t) = \max_{i \in p} fit_i(t), \quad p = 1, 2, \ldots, N \tag{14}$$

3.3. Gravitational constant (G) computation

Gravitational constant G is computed at iteration [8].

$$G(t) = G_o e^{(-\alpha t/T)} \tag{15}$$

Here, T is the maximum iteration, t is the current iteration and $\alpha > 0$ is the weight factor, computed as follows:

$$\alpha = \alpha_{max} - \frac{\alpha_{max} - \alpha_{min}}{T} \times t \tag{16}$$

3.4. Masses of the agents' calculation

Each agent's mass is calculated after computing current population's fitness as

$$M_{pi} = M_{ai} = M_{ii} = M_i, i = 1, 2, \ldots, N$$

Here, M_{pi} and M_{ai} are the active and passive gravitational masses respectively, while M_{ii} is the inertia of mass for ith agent.

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{17}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{18}$$

where $M_i(t)$ and $fit_i(t)$ represent the mass and the fitness value of the agent i at iteration t, respectively.

3.5. Velocity and positions of agents

The velocity and position of the agents are updated as

$$V_i^d(t+1) = \beta\, v_i(t) + a_i^d(t) \tag{19}$$

$$x_i^d(t+1) = x_i(t) + V_i^d(t+1) \tag{20}$$

Here β is the random number, $0 \le \beta \le 1$ and an acceleration of the ith agents at iteration 't' is computed as

$$a_i^d(t) = F_i^d(t)/M_i(t) \tag{21}$$

$F_i^d(t)$ is the total force acting on ith agent calculated as

$$F_i^d(t) = \sum_{j \in k_{best}\, j \neq i} \beta F_{ij}^d(t) \tag{22}$$

K_{best} is the set of first K agents with the best fitness value and biggest mass, which is a function of time, initialized to k_0 at the beginning and decreasing with time. Here k_0 is set to N (total number of agents) and is decreased linearly to 1.

$F_{ij}^d(t)$ is computed using the following equation:

$$F_{ij}^d(t) = G(t).\left(M_{pi}(t) \times \frac{M_{aj}(t)}{dis_{ij}(t)} + \varepsilon\right).\left(X_j^d(t) - X_i^d(t)\right) \tag{23}$$

Here x_i and X_j are the position vector of the ith and jth agent in dth dimension, $F_{ij}^d(t)$ is the force acting on agent i from agent j at dth dimension and ith iteration. $dis_{ij}(t)$ is the Euclidian distance between two agents i and j at iteration t. $G(t)$ is the computed gravitational constant at the same iteration while ε is a small constant. $M_{pi}(t)$ is the passive gravitational mass of the agent i at the instance t. $M_{aj}(t)$ is the active gravitational mass of the agent j at time t, these masses are calculated according to [20–23].

4. Novel hybrid particle swarm optimization and gravitational search algorithm (IPSO–IGSA)

The characteristics of all the population based meta-heuristic algorithms such as PSO and GSA are maintaining compromise between exploitation and exploration in order to solve the optimization problems. In PSO, p_{best} and g_{best} provides the exploration and exploitation. But, in GSA, the exploration can be guaranteed by choosing of the suitable value of the parameters such as Go and alpha and exploitation can be guaranteed only by reducing the participant agents. However, the PSO have ability for exploring in the multidimensional space, but GSA has its local exploitation capability.

There different ways of hybridization of PSO and GSA, one way of hybridization is serial mode i.e output of PSO/GSA as input to GSA/PSO. However, the result of the hybridization depends on the winning result of PSO and GSA. No such good result is generated in our work. Thus, another of hybridize PSO and GSA is consider based on the principle of any particle in the swarm as a particle introduced by PSO and/or GSA by means of applying co-evolutionary technique. Each particle in the hybridization process updates its position with the contributions of both PSO velocity and GSA acceleration. This hybrid algorithm called IPSO–GSA, proposed by employing cooperative behaviors of the particles affected by both GSA acceleration and PSO velocity to improve the performance and convergence of the algorithm. Thus, the velocity updating formulation in IPSO–GSA consists of cooperative contribution of PSO velocity and GSA acceleration.

$$V_i^d(t+1)_{PSO} = wV_i^d + C_1.\phi_1.(x_{pbest_i}^d - x_i^d(t)) + C_2.\phi_2.(x_{gbest}^d - x_i^d(t))$$

$$x_i^d(t+1)_{PSO} = x_i^d(t+1) + V_i^d(t+1) \tag{24}$$

$$V_i^d(t+1)_{GSA} = \beta\, V_i^d(t) + a_i^d(t) \tag{25}$$

where Eq. (24) is PSO velocity formulation is obtained from Eq. (2) and Eq. (25) is the GSA velocity formulation is obtained from Eq. (19)

$$V_i^d(t+1)_{HPSO-GSA} = C_3\phi_3 V_i^d(t+1)_{PSO} + C_4.(1-\phi_3).v_i^d(t+1)_{GSA} \tag{26}$$

$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1)_{HPSO-GSA} \tag{27}$$

where Eq. (26) is the IPSO–IGSA velocity formulation, which is formulated and updated using PSO velocity and GSA acceleration. In the IPSO–IGSA velocity equation, C_3 and C_4 are two acceleration coefficients that are used to adjust the PSO velocity and GSA acceleration on IPSO–IGSA. When C_3 or C_4 is set to zero, IPSO–IGSA becomes independently IPSO or IGSA and when C_3 and C_4 are set to one, IPSO–IGSA is stochastically influenced by half of IPSO and IGSA. ϕ_3 is a random variable generated within [0,1] which is determined the stochastic effect of the IPSO velocity and IGSA acceleration on the IPSO–IGSA velocity updating equation. Eq. (27) is used to update the position of particles in each iteration. In this hybridization process, IPSO uses a memory to save the best solution found so far, while IGSA uses the fitness value to adjust the accelerations. All the particles move very slowly when they are nearer to the good solution and *gbest* helps them to exploit the global best. Each particles observed the best solution (pbest and gbest) and moves towards it. Due to the above cause the hybridization of IPSO and IGSA is powerful to solve the optimization problem [42].

Procedure IPSO–IGSA (x^{curr_i}, y^{curr_i}, *pvector*)
Begin
Initialization: set of parameters $w_{min}, w_{max}, C_{1i}, C_{2i}, C_{1f}, C_{2f}, C_3, C_4, G_0, \alpha_{max}, \alpha_{min}$, *Max_Iter*, N
$X^{\rightarrow i} = (x^{curr\_i}, y^{curr\_i})$
Initialize particle with random positions, velocity and p_{best}
Evaluate fitness value of the initial particles
Evaluate g_{best}
For $k=1$ to *Max_Iter*
Begin
Compute α using Eq. (16)
Calculate the gravitational constant $G(t)$ using Eq. (15)
Compute the *worst*(t) and *best*(t) using Eqs. (13) and (14) respectively
For $i=1$ to N

Calculate $M_i(t)$ using Eq. (18)
Compute $dist_i$ using Eq. (6)
Compute w_i using Eq. (5)
Compute C_1 and C_2 using Eqs. (8) and (9) respectively

End For
For $i=1$ to N
For $d=1$ to *Number_of_dimension*
Calculate the acceleration $a_i^d(t)$ using Eq. (21)
Update the velocity $V_i^d(t)$ and position $_{xi}{}^d(t)$ using Eqs. (26) and (27)
End For
End For
For $i=1$ to N
Evaluate the fitness value $fit_i(t)$ of each particle
End For
Update $P_{best}(t)$ and $_{gbest}(t)$
End For
Update:
$x_i^{curr\_i} = x_j{}^{curr\_i} + V_i \cos\theta_i$
$y_i^{curr\_i} = y_j{}^{curr\_i} + V_i \sin\theta_i$
return
End

5. Problem formulation for multi-robot navigation

The multirobot navigation problem is formulated as to compute the next location for each robot from their current location in the environment by avoiding the collision with teammates(which is dynamic in nature) and obstacles (which are static in nature) in its path to reach at the goal. The set of principles is considered in formulating multi-robot path planning problem with the help of the following assumption.

5.1. Assumptions

1. Current position/initial position and goal position/target position of all the robot is known in prior coordinate system.
2. At any instant of time, the Robot can decide any action from a set of predefine actions for its motion.
3. Each robot is performing its action till reached in their respective target position in steps.

The following principles have been taken care for satisfying the given assumptions.

1. For determining the next position from its current position, the robot tries to align its heading direction towards the goal position.
2. The alignment may cause a collision with the robots/obstacles (which are static in nature) in the environment. Hence, the robot turns its heading direction with a certain angle either to left or right for determining its next position from its current position.
3. If a robot can align itself with a goal without collision, then, it will move to that determine the position.
4. If the heading direction is rotated to the left or right then it is required for the robot to rotate the same angle about its z-axis, if it is same for more than one, and then decide randomly.

Consider the initial position of the i^th robot at time t is (x_i^{curr}, y_i^{curr}), the next position of the same robot at time $(t+\Delta t)$ is (x_i^{next}, y_i^{next}), v_i^{curr} is the velocity of the robot R_i and (x_i^{goal}, y_i^{goal}) is the target or goal position of the robot .

So, the expression for the next position (x_i^{next}, y_i^{next}) can be derived from the Fig. 2 as follows:

$$x_i^{next} = x_i^{curr} + v_i^{curr} \cos\theta_i \Delta t \tag{28}$$

$$y_i^{next} = y_i^{curr} + v_i^{curr} \sin\theta_i \Delta t \tag{29}$$

When $\Delta t = 1$, the Eqs. (28) and (29) is reduced to

$$x_i^{next} = x_i^{curr} + v_i^{curr} \cos\theta_i \tag{30}$$

$$y_i^{next} = y_i^{curr} + v_i^{curr} \sin\theta_i \tag{31}$$

Consider initially, the robot R_i is placed in the location at (x_i^{curr}, y_i^{curr}). We want to find the next location of the robot

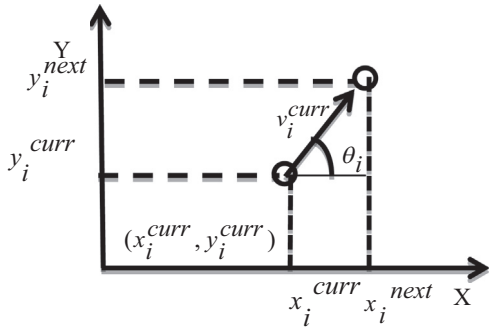

Fig. 2. Representation of next location from current location for ith robot.

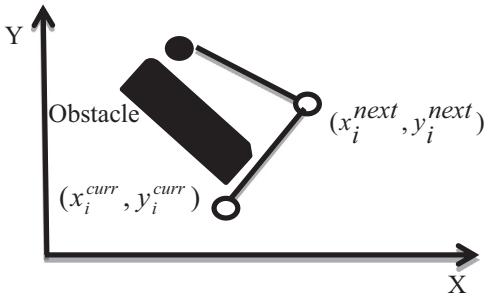

Fig. 3. Selection of next position (x_i^{next}, y_i^{next}) from current position (x_i^{curr}, y_i^{curr}) for avoiding collision with obstacle.

(x_i^{next}, y_i^{next}) joining of the two points between $\{(x_i^{curr}, y_i^{curr}), (x_i^{next}, y_i^{next})\}$ and $\{(x_i^{next}, y_i^{next}), (x_i^{goal}, y_i^{goal})\}$ should not touch the obstacle in the world map is represented in Fig. 3 and minimizes the total path length from current position to a goal position without touching the obstacle by forming constraint. Then objective function F_1 that determines the trajectory path length for n number of robots,

$$F_1 = \sum_{i=1}^{n} \left\{ \sqrt{((x_i^{curr} - x_i^{next})^2 + (y_i^{curr} - y_i^{next})^2)} + \sqrt{((x_i^{next} - x_i^{goal})^2 + (y_i^{next} - y_i^{goal})^2)} \right\} \tag{32}$$

By putting the value x_i^{next} and y_i^{next} from expressions (30) and (31) into expression (32), we obtain

$$F_1 = \sum_{i=1}^{n} \left\{ v_i^{curr} + \sqrt{\begin{matrix}(x_i^{curr} + v_i^{curr} \cos\theta_i - x_i^{goal})^2 \\ + (y_i^{curr} + v_i^{curr} \sin\theta_i - y_i^{goal})^2\end{matrix}} \right\} \tag{33}$$

The second objective function is considered as a repulsive function. The repulsive function is defined as a function of the relative distance between the robot and obstacles. Let $d_{min}(X_p)$ is the minimum distance of X_p from the obstacles. So the repulsive field for each static obstacle is defined in expression (34).

$$F_2(X_p) = \begin{cases} \frac{k}{\gamma}\left(\frac{1}{d_{\min}(X_p)} - \frac{1}{\eta_0}\right), & \text{if } d_{\min}(X_p) \le \eta_0 \\ 0\ , & \text{otherwise} \end{cases} \tag{34}$$

where η_0 is the influence range of the obstacle, k is positive constant and $\gamma \ge 2$ shapes the radial profile of potential. The third function is considered on the basis of prediction of the dynamic object in the world map, which will appear dynamically in the trajectory path of robots. So, the robot has to predict the dynamic obstacle position before deciding its next position for a moment. The objective function including the prediction principle is expressed as

$$F_3 = \sum_{i=1}^{n} \sqrt{(x_p - x_i^{goal})^2 + (y_p - y_i^{goal})^2} \tag{35}$$

Again, smoothness of the path is considered using fourth objective functions. The smoothness is expressed as the angle between two hypothetical lines connecting the goal point and two successive positions of the robot's in each iteration, i.e. $gbest^i$ and $gbest^{i-1}$ in ith iteration. The objective function for the smoothness of the path is expressed in mathematically as

$$F_4 = \frac{\cos^{-1}\left[\begin{matrix}(x_i^{curr} - x_i^{goal}).(x_{gbest\ i-1} - x_i^{goal}) \\ + (y_i^{curr} - y_i^{goal}).(y_{gbest\ i-1} - y_i^{goal}\end{matrix}\right]}{\sqrt{(x_i^{curr} - x_i^{goal})^2 + (y_i^{curr} - y_i^{goal})^2}} \times \sqrt{(x_{gbest\ i-1} - x_i^{goal})^2 + (y_{gbest\ i-1} - y_i^{goal})^2} \tag{36}$$

Now, the multirobot navigation problem can be represented as an optimization problem. The optimization problem contains an objective function that minimizes the Euclidean distance between the current location of each robot with their respective goal location and constraint based on avoiding collision with obstacles/teammates on their path. The constraints have been modeled by three types of penalty function. The first penalty function is used to avoid a collision of mobile robots with obstacles or teammates, whereas the second penalty is used to avoid collision between a mobile robot and dynamic obstacles and third penalty is considered for the smoothness of the path. Thus, the overall objective (or fitness) functions are obtained by the weighted sum of four objective functions such as

$$fit = \lambda_1 F_1 + \lambda_2 F_2 + \lambda_3 F_3 + \lambda_4 F_4 \quad (37)$$

where λ_1, λ_2, λ_3 and λ_4 are the weights of the shortest path, static obstacles, dynamic obstacles and smoothness of the path respectively. These weights are adjusted in the simulation and Khepera II robot, with preeminent values found $\lambda_1 = 0.25$, $\lambda_2 = 0.25$, $\lambda_3 = 0.25$ and $\lambda_4 = 0.25$. So, the optimized path is obtained by minimizing the fitness function in Eq. (37) with the assigned weights of each criterion.

6. Implementation of multi-robot path planning problem using hybrid IPSO–IGSA algorithm

Multi-robot path planning algorithm and flow chart is proposed in this section using hybrid IPSO–IGSA. The proposed hybrid IPSO–IGSA algorithm is used to evaluate the next positions of every robot by presuming the current positions of robots and speed as the parameter for optimizing the given multi-objective function. It determines the optimized path from each state to the goal state in a dynamic as well as static environment and robot measures its distance to obstacles with the help of equipped sensors. The flow chart for multirobot path planning using IPSO–IGSA is presented in Fig. 4.

The outline of the algorithm for multi-robot path planning is discussed below.

6.1. Pseudo-code for path planning

Input: (x_i^{curr}, y_i^{curr}), (x_i^{goal}, y_i^{goal}), and v_i^{curr} are the initial position, goal position and velocity for n robots respectively, $1 \le i \le n$ and ε is the threshold value.

Output: Optimal Trajectory of path OTP_i is generated for robot R_i from (x_i^{curr}, y_i^{curr}) to (x_i^{goal}, y_i^{goal})

Begin
For $i = 1$ to n
$x^{curr\_i} \leftarrow x_i^{curr}$; $y^{curr\_i} \leftarrow y_i^{curr}$;
End for
For each robot $i = 1$ to n
While $(Curr\_i \neq G_i)$ //$Curr\_i = (x_i^{curr}, y_i^{curr}), G_i = (x_i^{goal}, y_i^{goal})$//
Call IPSO–IGSA $(x^{curr\_i}, y^{curr\_i}, pvector)$;
// pvector is the position vector denotes updated current position of the i-th robot //
Moveto (x^{curr_i}, y^{curr_i});
End While
End for
End

7. Computer simulation

The path planning problem for multi-robot is carried out in a simulated environment. The simulation is conducted through programming in C language on a Pentium microprocessor and robot is represented with 14 similar soft-bots of circular shape with different color code. Each robot radius is 6 pixels. Prior to start of the experiment, the predefine initial location and goal location for all the robots is assigned. The experiments were conducted with seven differently shaped obstacles and assigned same velocities for each robot at the time of the program run; however, the velocities of each robot are adjusted in different runs of the same program. The initial configuration of the world map is our experimental result which is presented in Fig. 4 with seven obstacles and 12 soft-bots, out of 12 soft-bots Six are circular shape with different color represents the initial position of each robot and rest six are in rectangle shape represents the goal

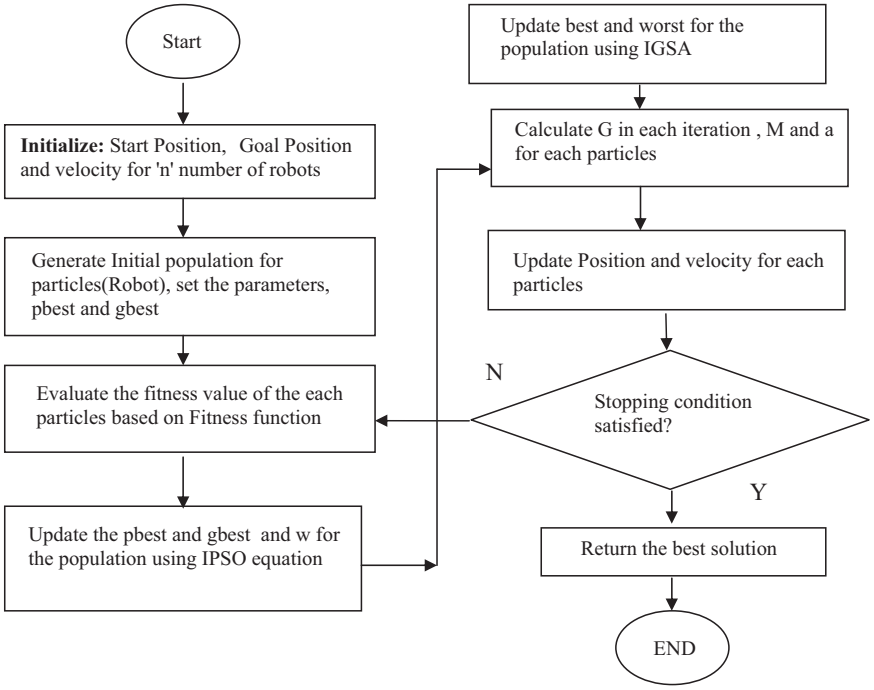

Fig. 4. Flow chart of multi-robot path planning By IPSO–IGSA.

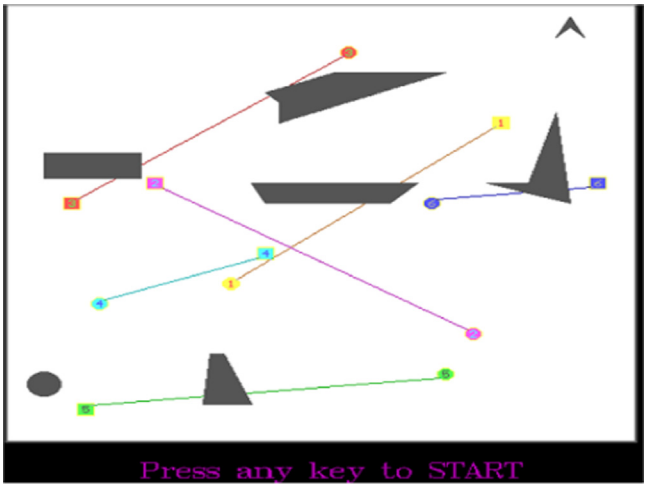

Fig. 5. Initial environment for 7 obstacles and 6 robots.

Table 1
Description of obstacles present in Fig. 5.

Obstacles	Position of obstacles
1	200,120,320,70,240,70,190,90,200,100
2	180,180,190,200,280,200,300,180,180,180
3	350,180,410,200,400,110,380,180,350,180
4	30,150,30,175,100,175,100,150,30,150
5	400,35,410,25,420,35,410,15,400,35
6	150,350,145,400,180,400,160,350,150,350
7	50,20,100,20,100,45,50,20

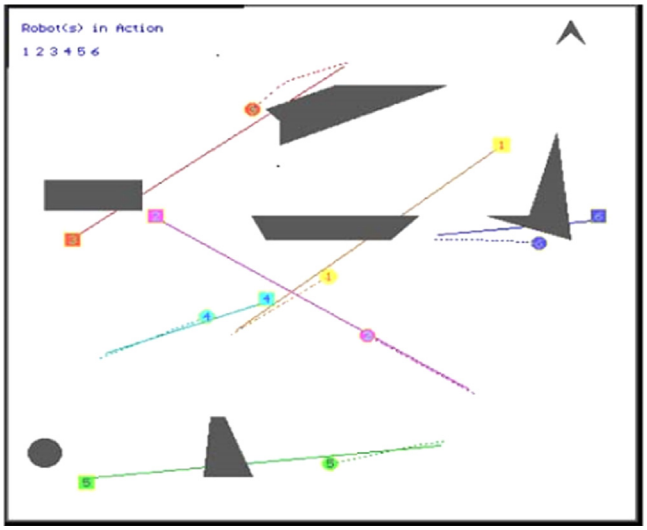

Fig. 6. Intermediate environment during execution of IPSO–IGSA after 9 steps.

position of each robot within the same color code. The positions of the obstacles present in Fig. 5 are described in Table 1. Figs. 6 and 7 present movements of the robots in the intermediate stages. The final stage of world map, where all the robots reached in their predefine goal respectively is shown in Fig. 8. This figure shows that the entire robot reached in their goal with 21 steps by IPSO–IGSA. The trajectory of path for IPSO and IGSA is presented in Figs. 9 and 10 respectively. All robots reached in their predefine goal in 26 and 31 steps respectively.

The optimal path generated from the simulation result for robots number from 1 to 6 is presented in Table 2. This table represents the initial co-oridnate, goal co-oridnate and sequence of optimal co-oridinate travelled during trajectory path planning

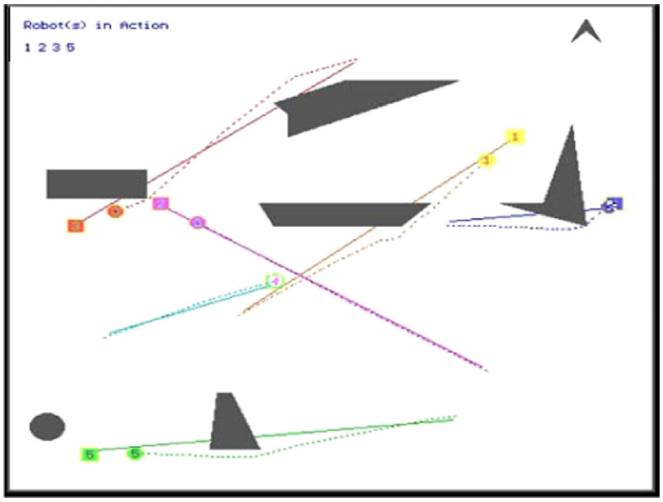

Fig. 7. Intermediate environment during execution of IPSO–EOPs after 17 steps.

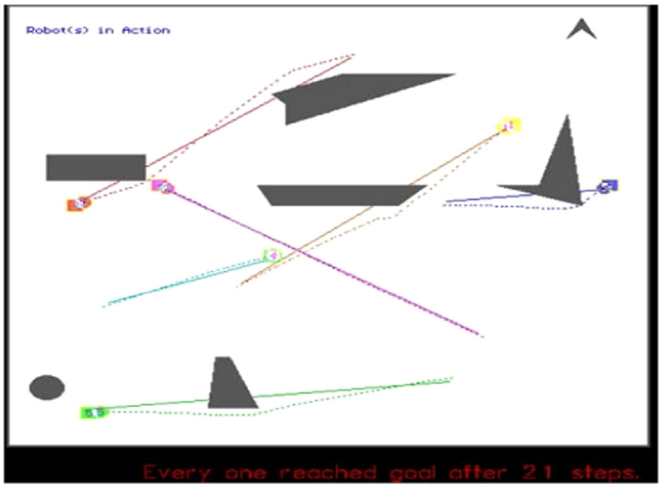

Fig. 8. All robots reached in their respective pre-defined goal in 21 steps by IPSO-EOPs.

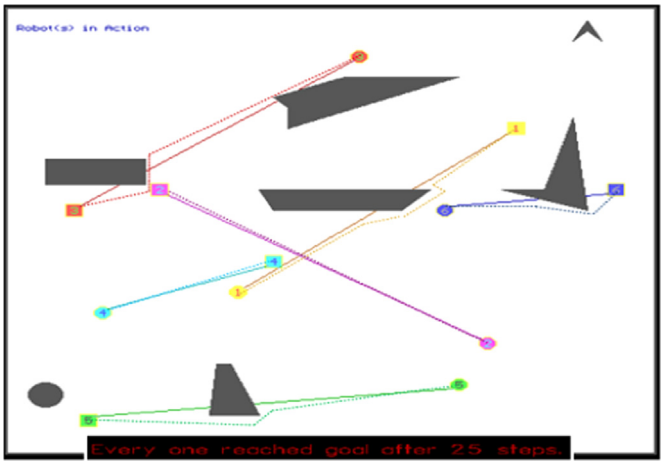

Fig. 9. All robots reached in their respective pre-defined goal in 26 steps by IPSO.

of the each robot in IPSO–IGSA. This shows that the path deviation in IPSO–IGSA is less in comparision to other two metaheriustics algorithm IPSO and IGSA irrespective of the robots. Here, for the shake of the complexity in simulation, we have consider for SIX number of robots and SEVEN obstacles. IPSO–IGSA worked for irrespective numbers of robot and shows that path deviation is

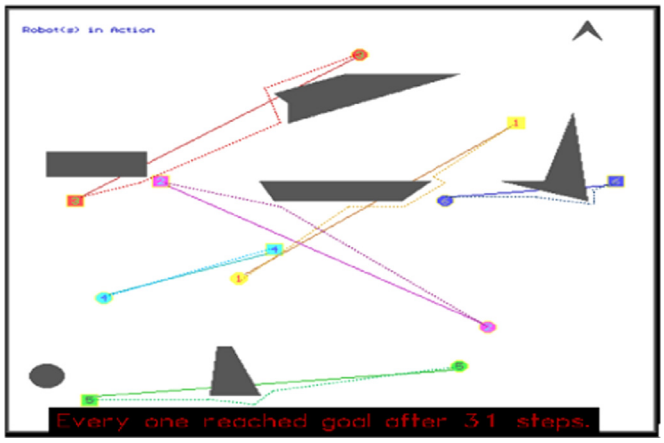

Fig. 10. All robots reached in their respective pre-defined goal in 31 steps by IGSA.

less in comparision to other algorithms. Number of optimal steps reqiured for different robots, number from 1 to 6 of the simulation result for different algorithm is presented in Table 3. Table 3 shows that the number of optimal steps required for IPSO–IGSA is less

Table 3
Number of steps required for robots from 1 to 6 to reach in goal.

Robot number	No of step required to goal in IPSO–IGSA	No of step required to goal in IPSO	No of step required to goal in IGSA
1	21	24	23
2	21	25	31
3	21	23	27
4	11	14	13
5	21	25	25
6	11	13	15

Table 2
Optimal trajectory of path from initial position to goal position for robots 1–6.

Robot Number	Initial position (in pixel)	Goal Position (in pixel)	Intermediate Sequence of Path from Initial Position to Goal Position in IPSO-IGSA (in pixel)
1	165,302	382,120	175,273 → 185,266 → 195,259 → 205,252 → 215,245 → 225,238 → 235,231 → 245,225 → 255,219 → 265,213 → 276,212 → 284,203 → 292,194 → 301,186 → 308,177 → 316,168 → 324,159 → 332,150 → 340,141 → 348,132 → **382,120**
2	340,330	110,180	329,323 → 318,316 → 307,309 → 296,302 → 285,295 → 274,288 → 263,281 → 252,274 → 241,267 → 231,260 → 220,253 → 210,246 → 199,239 → 188,232 → 178,225 → 167,218 → 157,211 → 146,204 → 136,197 → 125,190 → **110,180**
3	250,50	50,200	238,54 → 227,58 → 216,62 → 205,66 → 196,74 → 188,82 180,90 → 172,98 → 164,106 → 156,114 → 148,123 → 140,132 → 132,141 → 124,150 → 116,159 → 108,168 → 100,177 → 89,182 → 78,187 → 67,192 → **50,200**
4	70,300	190,250	81,295 → 92,290 → 103,285 → 114,280 → 125,275 → 136,270 → 147,265 → 158,261 → 169,257 → 180,253 → **190,250**
5	320,370	60,405	308,372 → 296,374 → 284,377 → 272,380 → 260,383 → 248,386 → 236,389 → 224,392 → 212,395 → 200,399 → 188,403 → 176,407 → 164,407 → 152,407 → 140,406 → 128,405 → 116,405 → 104,404 → 92,404 → 80,404 → **60,405**
6	310,200	430,180	321,200 → 332,200 → 343,200 → 354,201 → 365,201 → 376,202 → 387,203 → 398,203 → 409,200 → 417,191 → **430,180**

than the other algorithm such as IPSO and IGSA. The total number of optimal steps required for IPSO–IGSA, IPSO and IGSA is 21, 25 and 31 respectively.

The experiment is conducted using a central version of the algorithm using the fitness function (37) for deciding the next position of the robot. In our experiment, the following parameters are used in the simulation and Khepera II environment: dimension=2, No_of_particles (N)=50, $G_0=10$, *Max_Iter* (T)=30, $\alpha_{\min}=0.2$, $\alpha_{\max}=0.4$, $w_{\min}=0.4, w_{\max}=0.9, C_3=0.5$, $C_4=0.5$. In this proposed algorithm, C_1 is linearly decreased from $C_{1i}=2.5$ to $C_{1f}=0.5$ and C_2 is increased linearly from $C_{2i}=0.5$ to $C_{2f}=2.5$. The code is executed 30 times (maximum iteration) and the performance is analyzed based on the best result.

7.1. Average total trajectory path deviation (ATTPD)

Consider the robot R_k is generated a collision free trajectory of path from specified initial position nS_k to a goal position G_k in the jth number of iterations of the program is TP_{kj}. If $TP_{k1}, TP_{k2},\ldots,TP_{kj}$ are the collision free trajectory of paths produced in the jth number of iterations of the program. The average total trajectory path navigated (ATTPN) through jth number of iteration for the robot R_k is expressed as $\sum_{r=1}^{j} TP_{kr}/j$.The total average trajectory path deviation for the robot $R_k$is calculated by taking the difference between ATTPN and the real shortest path between S_k and G_k. If TP_{k-real} is the real trajectory path for robot R_k, then the total average trajectory path deviation is expressed by

$$ATTPD = TP_{k-real} - \sum_{r=1}^{j} TP_{kr}/j \tag{38}$$

Therefore, for n robots in the environment the average total trajectory path deviation (ATTPD) is

$$ATTPD = \sum_{i=1}^{n}\left(TP_{k-real} - \sum_{r=1}^{j} TP_{kr}/j\right) \tag{39}$$

7.2. Average untraveled trajectory target distance (AUTTD)

On a two dimensional workspace, the mathematical expression for the untraveled trajectory target distance for a given robot k in terms of specified goal position G_k and current position C_k is $\|G_k - C_k\|$, where $\|.\|$ denotes Euclidean norm. Similarly, untraveled trajectory target distance (UTTD)for n number of robots is UTTD$=\sum_{i=1}^{n}\|G_k - C_k\|$. We can calculate the average of UTTDs in the jth number of iterations as AUTTD$=\sum_{i=1}^{n}\|G_k - C_k\|/j$. The average untraveled trajectory target distance (AUTTD) with stages is presented in Fig. 11. It indicates that AUTTD takes more time to converge with decreasing the velocity and gradually terminated with iteration. Again, it is noted that larger the velocity of the robot,

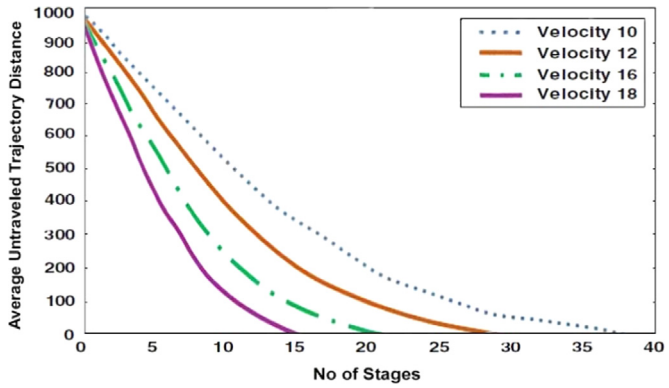

Fig. 11. Average untraveled trajectory distance vs. no. of stages with variable velocity and fixed number of obstacles=7.

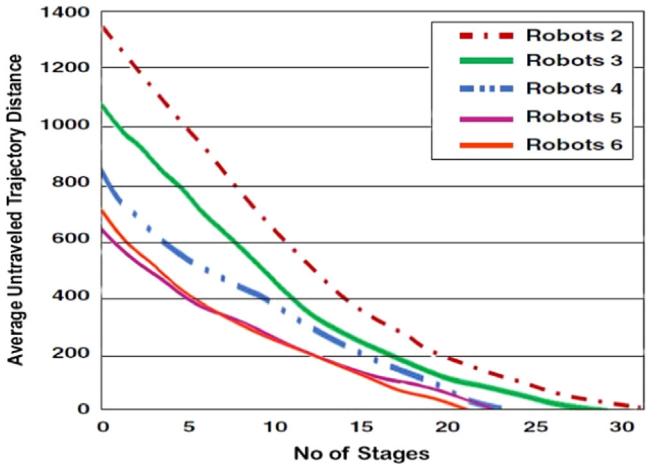

Fig. 12. Average untraveled trajectory distance vs. no. of stages with fixed obstacles and variable number of robots.

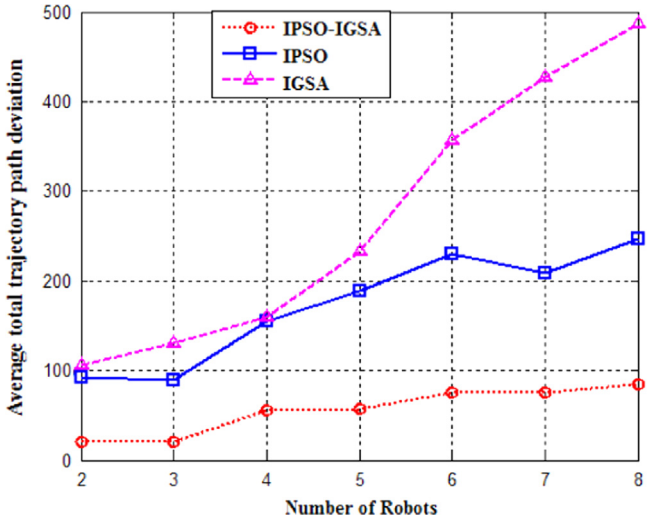

Fig. 13. Average total trajectory path deviation vs. no. of robots with variable no. of obstacles and velocity.

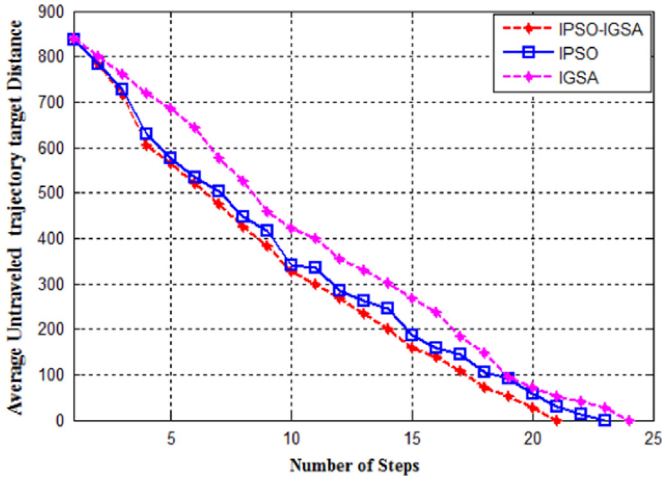

Fig. 14. No. of steps required in different algorithm vs. average untraveled trajectory target distance for respective algorithm.

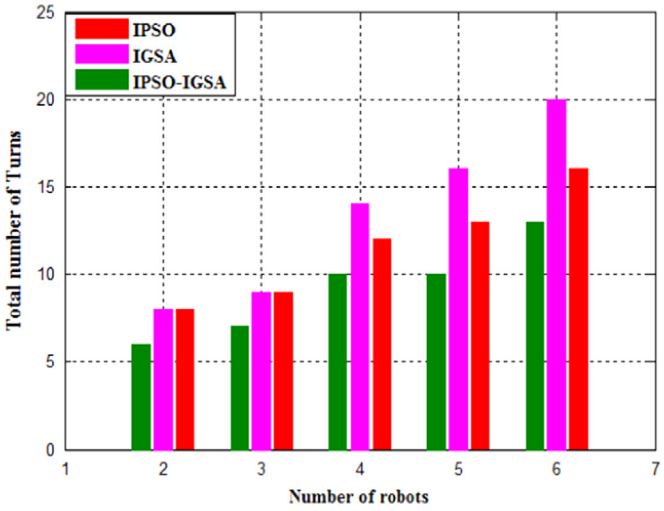

Fig. 15. Total number of turns vs. number of robots in three different algorithms.

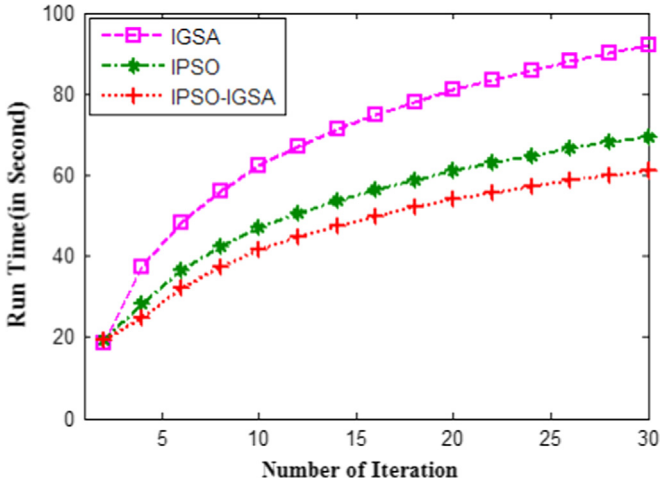

Fig. 16. Run time of IPSO–IGSA with number of iteration.

faster fall-off in the AUTTD. Fig. 12 shows that, with increasing the number of robots, slower the convergence rate. Slower the convergence causes the delay in fall-off in AUTTD. The performance of the result has been analyzed by plotting the average total trajectory path deviation (ATTPD) with the number of robots as variable in Fig 13. This path is generated by three different Evolutionary algorithms such as IPSO, IGSA, and IPSO–IGSA. Fig.13 shows the result of ATTPD computation, when the number of robots varies between 1 and 10. Here, we observed that IPSO–IGSA performs better than the other two algorithms as ATTPD is smallest for IPSO–IGSA in comparison to other two algorithms irrespective of the number of robots. The performance analysis has been performed in terms of AUTTD over the number of steps in Fig. 14. It provides graphs between AUTTD verse no stages required during the planning of path using three algorithms with number of obstacles=7 and no of robots=6. It is apparent from Fig.14 that AUTTD returns the smallest value for IPSO–IGSA irrespective of the number of planning steps. Now, the performance of the simulation result is analyzed in the terms of the number turn, by which we can able to minimize the energy consumption. The number of turn required for three different algorithm for number of robots=6 is demonstrate in Fig. 15. It shows that IPSO–IGSA takes less number of turn than other two algorithms and energy consumption to reach in the designation is less than other two algorithms. The simulation is only presented for six numbers of robots but number

Table 4
Simulation result of the three different algorithms in terms of average trajectory distance and time (s) to reach in goal.

Robot number	IPSO–IGSA		IPSO		IGSA	
	Avg. trajectory distance travelled	Time (s)	Avg. trajectory distance travelled	Time (s)	Avg. trajectory distance travelled	Time (s)
1	329.665	15.234	359.267	17.432	387.873	24.347
2	308.665	26.782	321.456	29.637	336.479	42.173
3	291.006	45.362	308.665	48.538	321.483	52.467
4	250.787	51.263	289.462	56.378	291.697	66.879
5	293.626	58.234	267.556	62.469	272.463	78.467
6	232.79	61.221	245.662	69.384	248.382	91.832

Table 5
Simulation result in terms of actual distance, average trajectory distance and standard deviation to reach in goal by IPSO–IGSA.

Benchmark	Generated path length		
	Actual distance	Avg. trajectory distance travelled	Std. dev
1 Robot	252.239	329.665	0
2 Robots	274.590	308.665	10.5
3 Robots	250	291.006	15.802
4 Robots	130	250.787	28.979
5 Robots	262.345	293.626	25.925
6 Robots	221.655	232.79	33.065

of turn is less for irrespective of the robot in the planning scheme of the algorithm. Finally, the performance analysis was undertaken by comparing the running time over the maximum number of iterations using three algorithms. Finally, the performance analysis was undertaken by comparing the running time over the number of iterations using three algorithms. Fig. 16 provides the time required for robots to reach in their respective goal position by three different algorithms and it shows that IPSO–IGSA takes less time for all robots to reach in their destination.

The result of the experiments generated through the proposed algorithm in the terms of the actual path travelled, average trajectory path travelled and standard deviation is presented in Table 4. Simulation result of the three different algorithms such as IPSO–IGSA, IPSO and IGSA in terms of Average trajectory, distance and time (in second) to reach in goal is presented in Table 5. For the simplicity of the environment, the experiment is conducted for Six numbers of robots and it can work on any number of robots. It shows that the proposed algorithm takes less time in comparison to the other two algorithms.

The result of the experiments performed are summarized in Table 6 in the terms of three performance metrics, namely, (1) total no of steps required to reach in the goal, (2) ATTPT and (3) ATTPD have been used here to determine the relative merits of IPSO–IGSA over the other algorithms for different robots. Table 6 confirms that outperforms the remaining two algorithms with respect to all three metrics for different robots.

7.3. Comparison study

In this subsection, the robustness of the proposed hybrid IPSO–IGSA algorithm has been verified by comparing with the existing PSO and GSA [58], hybrid PSO–GSA [30] and a new hybrid Honey Bee Mating Optimization-Tabu List [76] for multi-robot path planning. The authors [58] have implemented PSO and GSA algorithm for multi-robot path planning in Static Environments with Danger Zones. In their work, they formulated the objective function based on the maximization of distance between particles and danger source and

minimization of the distance between initial position and target position. They added a positive penalty value to the objective function in order to avoid the collision with the danger source. They investigated only the execution time for each algorithm PSO and GSA by considering different number of robots and number of iterations to reach at the goal. Similarly, the authors [30] have implemented hybrid PSO–GSA as an optimal path planning algorithm for multi-robot in static environment with danger zones. They formulated two objective functions and minimized these objective functions using PSO-GSA for generating collision free trajectory path in terms of minimizing the path length. They have not performed any analysis on various important aspects such as (1) The no of turns require for each robot to reach at the destination; (2) Total trajectory path deviation; (3) path uncovered by each robot in each iteration; (4) Energy utilization to reach the goal by each robot. The performance of the proposed IPSO–IGSA algorithm was verified by comparing the result obtained with those obtained by [58].

The simulation presented in Table 7. Shows that the proposed hybrid IPSO–IGSA is outperforms than PSO and GSA.

The authors [76] have proposed a hybrid algorithm for team robot motion planning in a dynamic environment. They have implemented honey bee mating optimization (HBMO) algorithm for minimizing robot travelling distance and tabu list technique for obstacle avoidance. Finally, the outcomes of the simulation result have been evaluated in the terms of average path deviation (ATPD) and average uncovered target distance (AUTD). In this paper, the

Table 6
Comparison of number of steps taken, ATTPT and ATTPD of different algorithms for different no. of robots.

No of robots	Algorithms (steps taken)			ATTPT			ATTPD		
	IPSO–IGSA	IPSO	IGSA	IPSO–IGSA	IPSO	IGSA	IPSO–IGSA	IPSO	IGSA
2	12	16	18	**15.19**	16.5	18.4	**5. 30**	6.7	7.7
3	15	18	20	**14.74**	18.6	20.4	**7. 26**	9.6	10.6
4	17	21	24	**16.75**	20.5	22.6	**18. 24**	20.3	21.9
5	19	24	26	**16.2**	21.6	24.7	**19.72**	21.43	23.89
6	**21**	**27**	**29**	**17.02**	**26.7**	**28.2**	**20.74**	**23.2**	**25.2**

Table 7
Comparison of simulation result of IPSO–IGSA in terms of no. of turns, execution time, no. of steps with Ref. [58].

No. of robots/no. iteration	Performance metrics	Existing proposed algorithm [58]		New proposed algorithm (IPSO–IGSA)		
		PSO	GSA	IPSO	IGSA	IPSO–IGSA
2/20	No of turns	NO	NO	9	9	8
	Execution time (s)	370	1410	32.564	38.783	28.735
	No of steps	NO	NO	18	21	14
2/30	No of turns			8	8	6
	Execution time (s)	515	2000	29.637	42.173	26.782
	No of steps	NO	NO	16	18	12
3/20	No of turns			9	10	8
	Execution time (s)	700	2300	52.583	56.284	48.749
	No of steps	NO	NO	20	21	17
3/30	No of turns			9	9	7
	Execution time (s)	1120	3300	48.538	52.467	45.365
	No of steps	NO	NO	18	20	15

Table 8
Comparison of simulation result of IPSO–IGSA with Ref. [76].

No. of obstacles	No. of robots	No. turn		No. of steps		ATTPD		AUTTD	
		Existing proposed algorithm (HMBO-Tabu list)	New proposed algorithm (IPSO–IGSA)	Existing proposed Algorithm (HMBO-Tabu list)	New proposed algorithm (IPSO–IGSA)	Existing proposed algorithm (HMBO-Tabu list)	New proposed algorithm (IPSO–IGSA)	Existing proposed algorithm (HMBO-Tabu list)	New proposed algorithm (IPSO–IGSA)
10	5	No.	**10**	22	**19**	23.87	**20.74**	38.56	**36.89**
20	5	No.	**12**	24	**19**	26.42	**23.56**	45.5	**42.24**
30	5	No.	**14**	26	**21**	28.54	**25.47**	53.69	**51.56**
10	10	No.	**12**	29	**25**	30.05	**27.23**	88.22	**86.12**
20	10	No.	**15**	31	**27**	31.23	**28.53**	104.1	**102.67**
30	10	No.	**17**	31	**27**	31.56	**28.89**	122.83	**120.23**
10	15	No.	**14**	32	**29**	33.87	**31.56**	201.82	**198.45**
20	15	No.	**16**	33	**29**	52.57	**51.34**	238.15	**234.34**
30	15	No.	**17**	33	**29**	73.7	**70.42**	281.02	**286.78**
10	20	No.	**15**	35	**31**	155.54	**152.45**	461.73	**458.76**
20	20	No.	**18**	35	**31**	189.49	**187.48**	554.84	**552.56**
30	20	No.	**20**	35	**31**	228.42	**224.67**	642.91	**640.34**
10	25	No.	**17**	41	**39**	378.29	**375.28**	1056.33	**1054.28**
20	25	No.	**20**	41	**39**	441.76	**439.29**	1246.47	**1244.78**
30	25	No.	**22**	41	**39**	513.48	**512.34**	1470.84	**1468.98**

Fig. 17. The Khepera II robot.

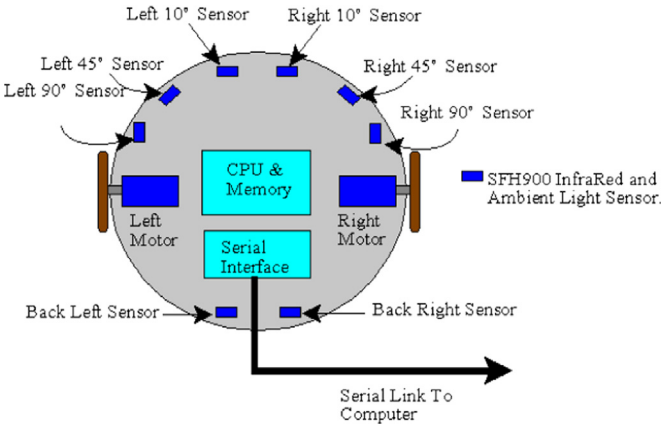

Fig.18. Position of sensors and internal structure of Khepera II.

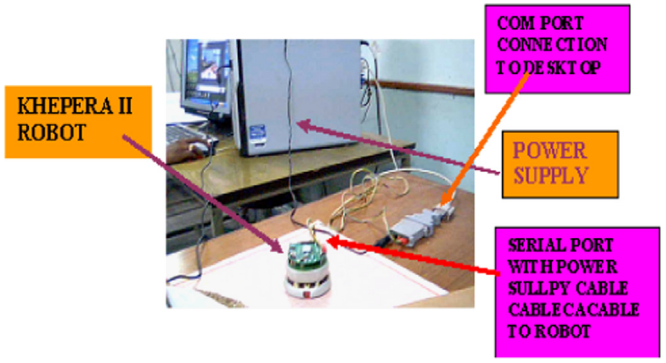

Fig. 19. Khepera network and its accessories.

authors have not evaluated the energy requirement for each robot, run time and not presented the simulation result for robot planning path. The comparison of results obtained from the proposed algorithm (IPSO–IGSA) with those obtained from HBMO-Tabu list has been presented in Table 8.

The comparison results presented in Table 8 shows that the proposed hybrid IPSO–IGSA is outperforms than HBMO-Tabu list [76].

8. Experiment on Khepera II robot

Khepera II is a miniature robot of diameter of 7 cm equipped with 8 in built infrared light sensors, and 2 relatively accurate encoders for the two motors control is shown in Fig. 17. The sensors are positioned

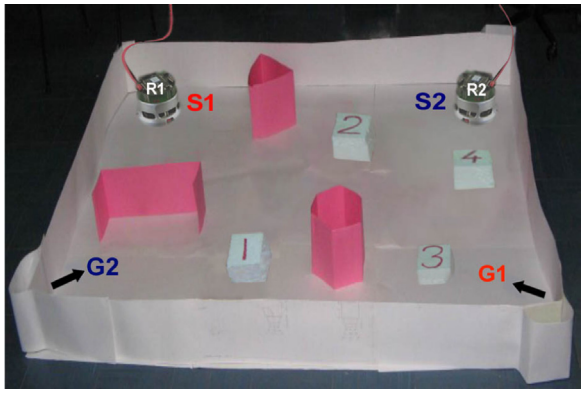

Fig. 20. Khepera environment setup for multi-robot path planning.

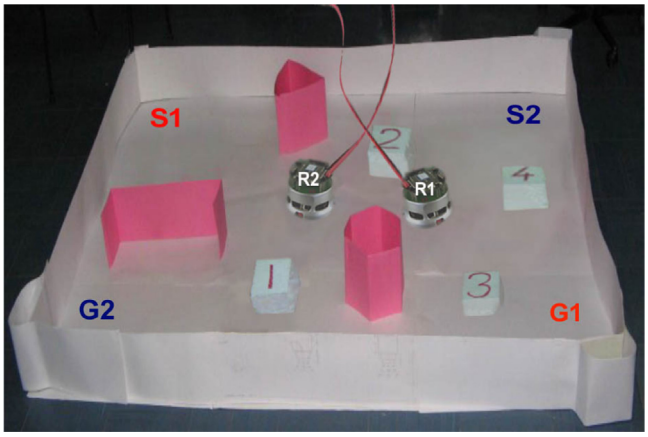

Fig. 21. Snapshot of Intermediate stage of multi-robot path planning using IPSO–IGSA in Khepera environment.

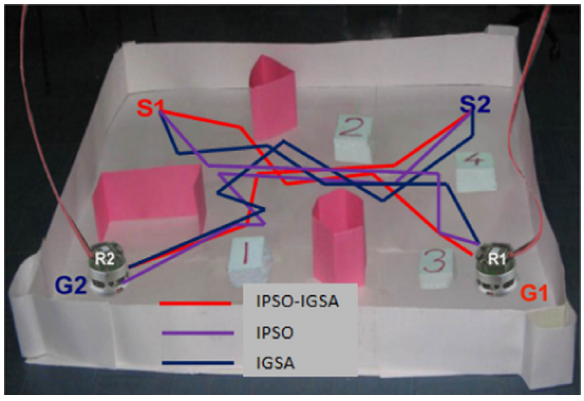

Fig. 22. Optimal path representation of different algorithm for mult-robot path planning in Khepera environment is represented by different color code. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

at fixed angles and have limited range detection capabilities. The sensors are numbered clockwise starting from the leftmost sensor 0 to sensor 7 and its internal structure is presented in Fig.18. Sensor values of the robot are measured in a numerical ranging from 0 to 1023. The value of the sensor is 1023, if the obstacle is approximately 2 cm distance from the robot and its value is zero, when the obstacle is more than 5cm from the robot. The robot consists of on board Motorola 68331 25 MHz microprocessor with a flash memory size of 512 kB. We used Khepera as a table-top robot and connected to a workstation through a wired serial link. The Khepera II network and its accessories are presented in Fig. 19 for conduct of experiment.

The initial world map for conducting the experiment in the Khepera II is presented in Fig. 20 with 8 obstacles of different shape and predefine initial state and goal is marked on the map, where different meta heuristic algorithm is applied. Fig. 21 shows the intermediate moment of the robot in the trajectory path towards the goal by respective robot using IPSO–IGSA.

Finally, different mata-heuristic algorithm is applied in Khepera environment and result of the trajectory path is presented in Fig. 22. IPSO–IGSA is implemented in the Khepera-II environment with considering two robots and compared with a different evolutionary computing algorithm is demonstrated in Fig. 21. The IPSO–IGSA implementation in Khepera-II shows better optimization in comparing to the other meta-heuristic algorithm such as IPSO and IGSA in the terms of the number of turns and path travelled. In the case of IPSO–IGSA, three numbers of turn are required to reach in the destination where as in the DE and IPSO required more no of turns. This shows that IPSO–IGSA is performing better than IPSO and IGSA.

9. Conclusion and future works

A hybridization of IPSO–IGSA algorithm was proposed for trajectory path planning of multi-robots in order to find collision free smoothness optimal path from predefine start position to end position for each robot in the environment. The results obtained from the experimental work are in good agreement with proposed algorithm. The performance of the proposed algorithm is compared with the different meta-heuristic algorithms such as IGSA, IPSO through the simulation and Khepera-II environment, it is concluded that the IPSO–GSA technique is best over other algorithms used for navigation of multi-mobile robot. However, in this paper, both the environment and obstacles are static relative to the robot; whereas other robots are dynamic for priority robots. Finally, the simulation results have compared with existing algorithm PSO and GSA, hybrid PSO–GSA and hybrid HMBO-Tabu list. In the future, work will be carried out using dynamic obstacles other than robots such as running vehicle, animals and onboard camera during the multi-robot path planning.

References

[1] M. Gerke, H. Hoyer, Planning of optimal paths for autonomous agents moving in homogeneous environments, In: Proceedings of the 8th International Conference on Advanced Robotics, 1997, pp. 347–352.

[2] J. Xiao, Z. Michalewicz, L. Zhang, K. Trojanowski, Adaptive evolutionary planner/navigator for mobile robots, IEEE Trans. Evol. Comput. l (1) (1997) 18–28.

[3] Z. Bien, J. Lee, A minimum-time trajectory planning method for two robots, IEEE Trans. Robot. Autom. 8 (3) (1992) 443–450.

[4] K.G. Shin, Q. Zheng, Minimum time collision free trajectory planning for dual robot systems, IEEE Trans. Robot. Autom. 8 (5) (1992) 641–644.

[5] I. Duleba, J.Z. Sasiadek, Nonholonomic motion planning based on Newton algorithm with energy optimization, IEEE Trans. Control. Syst. Technol. 11 (3) (2003) 355–363.

[6] M. Moll, L.E. Kavraki, Path-planning for minimal energy curves of constant length, In: Proceedings of the IEEE International Conference on Robotics & Automation, 2004, pp. 2826–2831.

[7] S.J. Buckley, Fast motion planning for multiple moving robots, In: Proceedings of the IEEE International Conference on Robotics & Automation, 1989, pp. 1419–1424.

[8] Xiang Liu, Daoxiong Gong, A comparative study of A-star algorithms for search and rescue in perfect maze, In: Proceedings of the International Conference on Electric Information and Control Engineering (ICEICE), 2011, pp. 24–27.

[9] Ma Changxi, Diao Aixia, Chen Zhizhong, Qi Bo, Study on the hazardous blocked synthetic value and the optimization route of hazardous material transportation network based on A-star algorithm, In: Proceedings of the 7th International Conference on Natural Computation, vol. 4, 2011, pp. 2292–2294.

[10] Xianmin Wei, Robot path planning based on Simulated Annealing & Artificial Neural networks, Res. J. Appl. Sci. Eng. Technol. 6 (1) (2013) 149–155.

[11] P.K. Das, S.K. Pradhan, S.N. Patro, B.K. Balabantaray, Artificial immune system based path planning of mobile robot, Soft Comput. Tech. Vis. Sci. Springer SCI 395 (2012) 195–207.

[12] Z. Zhan, J. Zhang, Y. Li, H.S. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B 39 (6) (2009) 1362–1381.

[13] G.-C. Luh, W.-C. Cheng, Behavior-based intelligent mobile robot using immunized reinforcement adaptive learning mechanism, Adv. Eng. Inf. 16 (2) (2002) 85–98.

[14] Amit Konar, Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain, 1st edition, CRC Press, 1999.

[15] P.K. Das, A. Konar, R. Laishram, Path planning of mobile robot in unknown environment, Int. J. Comput. Commun. Technol. 1 (2–4) (2010) 26–31.

[16] Preetha Bhattacharjee, Pratyusha Rakshit, Indrani Goswami, Amit Konar, Atulya K. Nagar, Multi-robot path-planning using artificial bee colony optimization algorithm, In: Proceedings of the IEEE Congress on Nature and Biologically Inspired Computing, 2011, pp. 219–224.

[17] Amit Jayasree Chakraborty, Lakhmi C. Konar, Uday Kumar Jain, Chakraborty, Cooperative multi-robot path planning using differential evolution, J. Intell. Fuzzy Syst. 20 (1–2) (2009) 13–27.

[19] R. Regele, P. Levi, Cooperative multi-robot path planning by Heuristic priority adjustment, In: Proceedings of the IEEE/RSJ International Conference on Robotics System, 2006, pp. 5954–5959.

[20] Y. Zhang, D.-W. Gong, J.-H. Zhang, Robot path planning in uncertain environment using multi-objective particle swarm optimization, Neurocomput 103 (2013) 172–185.

[21] E. Masehian, D. Sedighizadeh, Multi-objective PSO-and NPSO based algorithms for robot path planning, Adv. Electr. Comput. Eng. 10 (2010) 69–76.

[22] Y. Guo, L.E. Parker, A distributed and optimal motion planning approach for multiple mobile robots, In: Proceedings of the IEEE International Conference on Robotics and Automation, vol. 3, 2002, pp. 2612–2619.

[23] A. Ratnaweera, S.K. Halgamuge, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficient, IEEE Trans. Evol. Comput. 83 (2004) 240–255.

[24] Swagatam Das, Amit Konar, Uday Kumar Chakraborty, Improving particle swarm optimization with differentially perturbed velocity, In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, 2005, pp. 177–184.

[25] Mohammad Javad Mahmoodabadi, Z. Salahshoor Mottaghi, and Ahmad Bagheri. "HEPSO: high exploration particle swarm optimization, Inf. Sci. 273 (2014) 101–111.

[26] J.J. Liang, Ak Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 103 (2006) 281–295.

[27] Rupam Kundu, Rohan Mukherjee, Swagatam Das, Modified particle swarm optimization with switching update strategy, In: Proceeding of the Swarm, Evolutionary and Memetic Computing (SEMCCO), 2012, pp. 644–652.

[28] M.H. Sadafi, R. Hosseini, H. Safikhani, A. Bagheri, M.J. Mahmoodabadi, Multi-objective optimization of solar thermal energy storage using hybrid of particle swarm optimization, multiple crossover and mutation operator, Int. J. Eng. Trans. B 243 (2011) 367–376.

[29] M. Lovbjerg, T.K. Rasmussen, T. Krink, Hybrid particle swarm optimizer with breeding and subpopulations, In: Proceedings of the Genetic, Evolutionary Computation, 2001, pp. 469–476.

[30] C. Purcaru, R.-E. Precup, D. Iercan, L.-O. Fedorovici, R.-C. David,and, F. Dragan, Optimal robot path planning using gravitational search algorithm, Int. J. Artif. Intell. 10 (2013) 1–20.

[31] P. Bhattacharya, M.L. Gavrilova, Roadmap-based path planning using the Voronoi diagram for a clearance-based shortest path, IEEE Robot. Autom. Mag. 15 (2) (2008) 58–66.

[32] B.K. Oleiwi, H. Roth, B.I. Kazem, A hybrid approach based on ACO and GA for multi objective mobile robot path planning, Appl. Mech. Mater. 527 (2014) 203–212.

[33] Constantin Purcaru, R.-E. Precup, Daniel Iercan, L.-O. Fedorovici, R.-C. David, Hybrid PSO-GSA robot path planning algorithm in static environments with danger zones, In: Proceedings of the 17th International IEEE Conference on System Theory, Control and Computing (ICSTCC), 2013, pp. 434–439.

[34] Ming-Yi Ju, Siao-En Wang, Jian-Horn Guo, Path planning using a hybrid evolutionary algorithm based on tree structure encoding, Sci. World J. (2014).

[35] P.J. Angeline, Using selection to improve particle swarm optimization, In: Proceedings of the IEEE Congress on Evolutionary Computation, 1998, pp. 84–89.

[39] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (13) (2009) 2232–2248.

[40] Om. Prakash Verma, Rishabh Sharma, Manoj Kumar, Neetu Agrawal, An optimal edge detection using gravitational search algorithm, Lect. Notes Softw. Eng. 1 (2) (2013) 148–152.

[41] Norlina Mohd Sabri, Mazidah Puteh, Mohamad Rusop Mahmood, A review of gravitational search algorithm, Int. J. Adv. Soft Comput. Appl. 5 (3) (2013) 1–39.

[42] S. Mirjalili, S.Z. Mohd Hashim, H. Moradian Sardroudi, Training feed-forward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Appl. Math. Comput. 218 (111) (2012) 25–37.

[43] Mansoor Davoodi, et al., Clear and smooth path planning, Appl. Soft Comput. 32 (2015) 568–579.

[44] Xin Chen, Yangmin Li, Smooth path planning of a mobile robot using stochastic particle swarm optimization, In: Proceedings of the IEEE International Conference on Mechatronics and Automation, 2006, pp. 1722–1727.

[45] Dun-wei Gong, Jian-hua Zhang, Yong Zhang, Multi-objective particle swarm optimization for robot path planning in environment with danger sources, J. Comput. 6 (8) (2011) 1554–1561.
[46] D.B. Chen, C.X. Zhao, Particle swarm optimization based on endocrine regulation mechanism, J. Control. Theory Appl. 24 (6) (2007) 1005–1009.
[47] Q.R. Zhang, G.C. Gu, Path planning based on improved binary particle swarm optimization algorithm, In: Proceedings of IEEE International Conference on Robotics, Automation and Mechatronics, 2008, pp. 462–466.
[48] S. Bashyal, G.K. Venayagamoorthy, Human swarm interaction for radiation source search and localization, In: Proceedings of the IEEE Swarm Intelligence Symposium, 2008, pp. 1–8.
[49] Q. Yuan-Qing, S. De-Bao, L. Ning, C. Yi-Gang, Path planning for mobile robot using the particle swarm optimization with mutation operator, In: Proceedings of the International Conference on Machine Learning and Cybernetics, 2004 pp. 2473–2478.
[50] M. Hua-Qing, Z. Jin-Hui, Z. Xi-Jing, Obstacle avoidance with multiobjective optimization by PSO in dynamic environment, In: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 5, 2005, pp. 2950–2956.
[51] M. Saska, M. Macas, L. Preucil, L. Lhotska, Robot path planning using particle swarm optimization of ferguson splines, In: Proceedings of the IEEE/ETFA, vol. 06, 2006, pp. 833–839.
[52] C. Xin, L. Yangmin, Smooth path planning of a mobile robot using stochastic particle swarm optimization, In: Proceeding of the IEEE Conference on Mechatronics and Automation, 2006, pp. 1722–1727.
[53] Adham Atyabi, David M.W. Powers, Cooperative area extension of PSO transfer learning vs. uncertainty in a simulated swarm robotics, In: Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO), 2013, pp. 177–184..
[54] Adham Atyabi, Somnuk Phon-Amnuaisuk, Chin Kuan Ho, Navigating a robotic swarm in an uncharted 2D landscape, Appl. Soft Comput. 10 (2010) 149–169.
[55] Adham Atyabi, Somnuk Phon-Amnuaisuk, Chin Kuan Ho, Applying area extension PSO in robotic swarm, J. Intell. Robot Syst. 58 (3–4) (2010) 253–285.
[56] Adham Atyabi, David M.W. Powers, The use of area extended particle swarm optimization (AEPSO) in swarm robotics, In: Proceedings of the Eleventh International Conference on Control, Automation, Robotics and Vision, 2011, pp. 591–596.
[57] Chengyu Hu, Xiangning Wu, Qingzhong Liang, Yongji Wang, Autonomous Robot Path Planning Based on Swarm Intelligence and Stream Functions, ICES2007, LNCS 4684, Springer, 2007, pp. 277–284.
[58] Purcaru Constantin, et al., Multi-robot GSA-and PSO-based optimal path planning in static environments, In: Proceedings of the IEEE International Conference on Robot Motion and Control (RoMoCo), 2013, pp. 197–202.
[59] Pei Li, HaiBin Duan, Path planning of unmanned aerial vehicle based on improved gravitational search algorithm, Sci. China Technol. Sci. 55 (10) (2012) 2712–2719.
[60] Seyedali Mirjalili, Siti Zaiton Mohd Hashim, A new hybrid PSOGSA algorithm for function optimization, In: Proceedings of the 17th IEEE International Conference on Computer and Information Application (ICCIA), 2010, pp. 374–377.
[61] Xianhai Song, et al., Grey Wolf optimizer for parameter estimation in surface waves, Soil. Dyn. Earthq. Eng. 75 (2015) 147–157.
[62] Shanhe Jiang, Zhicheng Ji, Yanxi Shen, A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints, Int. J. Electr. Power Energy Syst. 55 (2014) 628–644.
[63] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, C. Lucas, Memetic algorithm based path planning for a mobile robot, Int. J. Inf. Technol. 1 (2004) 174–177.
[64] Y.N. Guo, M. Yang, J. Cheng, Path planning method for robots in complex ground environment based on cultural algorithm, In: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC'09), 2009, pp.185–192.
[65] Hongwei Mo, Lifang Xu, Research of biogeography particle swarm optimization for robot path planning, Neurocomputing 148 (2015) 91–99.
[66] K. Krishnanand, D. Ghose, A glowworm swarm optimization based multi-robot system for signal source localization, in: D. Liu, L. Wang, K. Tan (Eds.), Design and Control of Intelligent Robotic Systems, Vol. 177 of Studies in Computational IntelligenceSpringer, Berlin, Heidelberg, 2009, pp. 49–68.
[67] K. Krishnanand, D. Ghose, Chasing multiple mobile signal sources: a glowworm swarm optimization approach, In: Proceedings of the 3rd Indian International Conference on Artificial Intelligence (IICAI-07), 2007, pp. 1308–1327.
[68] Jinchao Guo, Yu Gao, Guangzhao Cui, The path planning for mobile robot based on bat algorithm, Int. J. Autom. Control. (1) (2015) 50–60.
[69] Prases K. Mohanty, Dayal R. Parhi., Optimal path planning for a mobile robot using cuckoo search algorithm, J. Exp. Theor. Artif. Intell. (2014) 1–18.
[70] Zexuan Zhu, et al., Global path planning of mobile robots using a memetic algorithm, Int. J. Syst. Sci. 46 (11) (2015) 1982–1993.
[71] Chang Liu, Zhongqiang Gao, Weihua Zhao, A new path planning method based on firefly algorithm, In: Proceedings of the Fifth International Joint Conference on IEEE International on Computational Sciences and Optimization (CSO), 2012, pp. 775–778.
[72] Shahrzad Saremi, Seyedeh Zahra Mirjalili, Seyed Mohammad Mirjalili, Evolutionary population dynamics and grey wolf optimizer, Neural Comput. Appl. (2014) 1–7.
[73] Xianbing Meng, et al., A new bio-inspired algorithm: chicken swarm optimization, Adv. Swarm Intellig. Springer Int. Publ. (2014) 86–94.
[74] Junpeng Li, et al., An improved krill herd algorithm: krill herd with linear decreasing step, Appl. Math. Comput. 234 (2014) 356–367.
[75] Gai-Ge Wang, Suash Deb, Zhihua Cui, Monarch butterfly optimization, Neural Comput. Appl. (2015) 1–20.
[76] Mohammad Abaee Shoushtary, Hasan Hoseini Nasab, Mohammad Bagher Fakhrzad, Team robot motion planning in dynamics environments using a new hybrid algorithm (honey bee mating optimization-tabu list), Chin. J. Eng. (2014) .