Applied Soft Computing Journal 86 (2020) 105929

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Multi robot distance based formation using Parallel Genetic Algorithm

A. López-González [a,*], J.A. Meda Campaña [a], E.G. Hernández Martínez [b], P. Paniagua Contro [b]

[a] Instituto Politécnico Nacional, SEPI-ESIME Zacatenco, Avenida IPN S/N, 07738 CDMX, Mexico
[b] Universidad Iberoamericana, DICAT, Prolongación Paseo de Reforma 880, 01219, CDMX, Mexico

ARTICLE INFO

Article history:
Received 27 June 2018
Received in revised form 16 July 2019
Accepted 4 November 2019
Available online 12 November 2019

MSC:
00-01
99-00

Keywords:
Parallel Genetic Algorithm
Multi robot
Consensus
Formation
Distributed Artificial Intelligence

ABSTRACT

In this paper an alternative method to achieve distance based formation is presented. The method uses Genetic Algorithms to find a suitable solution based on angle and distance, and an appropriate constant velocity to avoid collisions. The designed algorithm is extended to a parallel scheme to improve its performance and achieve Artificial Distributed Intelligence, in which the robots share, through solution migration, the best ways to converge to desired distances while avoiding collisions, finally reaching consensus on the solution. The algorithm is tested using simulations and real robots experiments.

1. Introduction

Researchers of the fast growing field of Multi robot systems (MRS) focus on the idea that there is strength in numbers, as presented in Dudek's example [1] of a task that requires simultaneous operation where the restrictions of a single robot system is overcome by a MRS.

MRS made out of a large number of simple and cheap robots could have better performance than individual highly specialized and costly robots, mainly because: with more robots exploration of a wider spatial area can be faster and distributed specialized jobs can be carry out by different team members if the MRS is heterogeneous. Another important characteristic of MRS is redundancy, a large number of robots support each other carrying out teammates task if the robots fails to achieve the mission [2].

There is a lot of research in MRS on designing appropriate coordination strategies and control laws that enable them to achieve objectives efficiently together [3–5].

Coordination strategies include various problems that aim to drive a group of agents to some common state, this is usually called consensus, agreement, synchronization or rendezvous, in this paper this strategies are referred as the *consensus problem*, extensively studied in [6,7], modelling robots with single integrator and using graph theory describe topology, ergo, connections between robots.

Control laws are designed to achieve consensus avoiding inter robot interference [8]. Global convergence, group stability, group following and other characteristics are studied through their topology graphs [2].

Some variables where robots can achieve consensus are, position, displacement or distance with regard to other robots, this kind of consensus is called formation control [9].

This consensus behaviours are supposed to be applied in exploration and mapping, security and surveillance tasks, biological systems simulations and others [4,10].

In our previous work, we studied the distance based formation [11,12] with area restrictions [13] and marching control in [14], using Lyapunov type strategies based on attractive/repulsive position or distance based potential functions, called error based gradient strategies.

The main disadvantage of this exact consensus strategies is the complexity of mathematical operations needed to achieve the aimed formation, this can be augment by the mathematical model of the robots and a large number of robots. Another approach to the formation problem is the use of metaheuristic algorithms or computational models. Particle swarm optimization (PSO) [15], Artificial neural networks [16], A* algorithm [17], Finite estates machines (FEA) [18], Ant colony optimization (ACO) [19] and many others have been used to achieve different types of formation. Most of the metaheuristic algorithms cannot guarantee

* Corresponding author.
E-mail addresses: alexandro.lopez@ibero.mx (A. López-González), jmedac@ipn.mx (J.A. Meda Campaña), eduardo.gamaliel@ibero.mx (E.G. Hernández Martínez), pablo.paniagua@correo.uia.mx (P.P. Contro).

https://doi.org/10.1016/j.asoc.2019.105929
1568-4946/

optimality or even a feasible solution, but their use is intended to simplify the mathematical burden of exact strategies.

Other disadvantage of exact strategies, like the error based gradient strategies, is that they are effective but isolated in each robot, most consensus strategies can be denominated as reaching decentralized agreement.

There are three levels of multi robot interaction, coordination, cooperation and collaboration [20]. Coordination occurs when robots are aware of each other and share a common space, working around each other to minimize mutual interference, hence their actions do not help other team members achieve their goals. Cooperation happens when robots share goals and their conjoint actions are beneficial to all teammates, so multiple robots work together and reason about each other's capabilities in order to accomplish a joint task. Collaboration arise when robots have individual goals, but they are aware of their teammates goals and capabilities, and their individual actions help advance their goals and others' goals.

In this context consensus strategies barely cooperate by communicating position, velocities and accelerations, these strategies mainly coordinate the robots.

Metaheuristic strategies are better at collaboration between robots. Collaboration in a MRS is achieved through Distributed Artificial Intelligence (DAI). The DAI field is mainly concerned with problems involving agents working together solve problems [21]. The advantages of distributing processes are flexibility and robustness, through redundancy and parallelism. The knowledge based DAI paradigm focuses on information sharing between robots, in order to allow them to use such awareness to solve group problems [20].

One of the most important and powerful algorithm used is the metaheuristic Genetic algorithms (GAs) [22–25]. A GA is a arbitrary search and optimization algorithm which mimics biological evolution. Introduced in 1975 by Professor Holland [26], GAs have characteristics of powerful self adaptive capability and global parallel optimization [27,28].

Because evolution is a highly parallel process, one of the greatest strengths of GAs is parallelism [29]. Parallel Genetic Algorithms (PGAs) [30,31] are being used to solve hard problems. As the complexity of a problem increases, GAs need a bigger population, this translates directly into higher computational costs. The logic behind PGAs is to reduce the processing time needed to reach an acceptable solution. Comparing serial GAs to PGAs, using different parallel architectures, better solutions can be found, in less time, with these latter [32,33]. As stated in [34] the most important characteristic of DAI is combining the computational resources of the group augmenting group intelligence over individual agents capabilities. This is very related to PGAs, using the parallel computational power of many agents might improve the solution for the group objectives.

2. Related work

While metaheuristic algorithms are widely used for multi robot path planning [35,36], in the field of MRS, GAs and PGAs have been used scarcely to achieve formation. This fact is due to the distinction of deliberative and reactive schemes for MRS formation [37].

Many works use GAs for parameter tunning [38]. In [39] authors exploit these algorithms to tackle the multi robot coalition formation problem, using multiobjective PGAs. A Voronoi global planer is mixed with a GA based on artificial potential models in [40] to achieve motion planning of robot swarms. Genci Capi and Zulkifli Mohamed [41] used GAs to create artificial neural networks that control each robot in the formation.

Kobayashi, Tomita and Kojima in [42] present a reformation algorithm based on GAs, with an obstacle avoidance achieved through Q-Learning. The algorithm presented acts as a reactive algorithm that maintains formation using distance rules, while avoiding collisions. Nevertheless, the robots' initial conditions start at the desired formation, and the algorithm is isolated in each robot, there is not any communication between robots.

Nazarahari, Khanmirza and Samira Doostie in [43] use an hybrid GA with potential fields to solve the multi robot path planning problem. The algorithm is tested using simulations on different scenarios, and compared to other search algorithms. While this scheme approaches to the presented work, the GA's are used to optimize the original potential field trajectories. It is also a centralized algorithm without distribution amongst robots.

As depicted in [44,45] PGAs have been used for fast path planning, with excellent results, but not as distributed intelligence for MRS. The parallel processes allow improvement in convergence, beyond that this paper intends to demonstrate that it could also be used as DAI in MRS [46].

3. Research objectives

The objective of this research is to evaluate the use of metaheuristics as fast reactive control in MRS. In MRS metaheuristics are usually used as planners for paths, task distribution or process optimization, mainly as deliberative strategies. But with the recent advances in microcontrollers, microcomputers and communication devices, fast calculations of metaheuristic algorithms (non exact strategies) might prove better than mathematically dense exact strategies. The main hypothesis is that a MRS could combine the processing power of all agents in order to solve in real time the consensus problem, reaching DAI.

The central contribution of this paper is a distance based formation control PGA. The main features of the approach are the next points:

- Distance based formation is achieved using distance and angle, while collision avoidance is achieved through velocity control. Using GAs and PGAs as control laws not path planing algorithms.
- The distance based formation GA is conceived as a parallel algorithm, robots share their own solutions, improving solutions and accelerating convergence time for all robots, hence it can be considered as DAI.
- This approach works for n robots, modelled as single integrators. It becomes a general result, because distinct nonholonomic robots can be moved using an appropriate input/output linearization [11].
- The approach is validated in appropriate numerical simulations and real time experiments.

The rest of this paper is organized as follows, Section 4 describes the MRS distance based formation problem, describes the main algorithm proposed and Sections 4.1 and 4.2 present simulation results for the simple and parallel schemes respectively, Section 4.3 explains the main results presented using real robots, all results are analysed on Section 5, finally Section 6 explains the conclusions and work ahead.

4. Distance based formation using GA

MRS distance based formation problem is defined as follows, consider $N = [R_1, \ldots, R_n]$ as a group of mobile robots with Cartesian positions $\xi = \xi_1, \ldots, \xi_n$, with $\xi_i = [x_i, y_i]$, $i = 1, \ldots, n$ respectively. An elementary mathematical model for a robot is given by

$$\dot{\xi}_i = u_i \in \mathbb{R}^2, \quad i = 1, \ldots, n \tag{1}$$

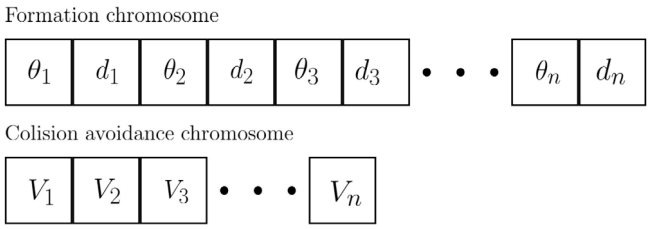


Fig. 1. GA chromosomes.

with u_i being the vector velocity for the robot R_i. Eq. (1) introduces the single integrator mobile robot model, that helps to simplify the formation control laws analysis [6,9].

A distance based formation graph is given by $G = \{Q, E, D\}$, where $Q = \{R_1, R_2, \ldots, R_n\}$ refers to the vertices, $E = \{(j, i) \in Q \times Q\}$ includes the edges formed by pairs of nodes that represent inter robots connections, hence $(j, i) \in E$ if $j \in N_i$, and $D = \{d_{ji}\}$, $\forall (j, i) \in E$ defines the desired distances between robots i and j, i.e. $\|\xi_i - \xi_j\| = d_{ij} \in \mathbb{R}$, $\forall i \neq j, j \in N_i$ in a desired formation pattern.

A well defined distance based formation graph must satisfy that $d_{ij} = d_{ji}$ for all $(j, i) \in E$, and the graph must be connected, i.e. there are no isolated nodes.

The proposed GA achieves two objectives, distance based formation and collision avoidance using the proposed formation chromosome for solving the necessary trajectory is shown in Fig. 1. The formation chromosome is composed by N vectors, described with angle and distance indicating the suggested motion for every robot in order to achieve the distance based formation between robots, hence, the robots must execute linear trajectories to reach formation. The validity of each trajectory is evaluated using distance error between robots. Fig. 2a depicts two point robots with initial positions P_1 and P_2 moving at angles θ_1 and θ_2, distances d_1 and d_2 to achieve a final distance between robots d_{12}.

Due to the linear trajectories proposed to complete distance based formation collisions can occur. Collision point depends on the vector magnitude and angle, but also on the robot dimension. To avoid collisions on the linear path, the velocities for each robot are calculated using the collision avoidance chromosome shown in Fig. 1. Given a safety radius r_n for each robot, a robot velocity is calculated comparing, the position along the vector trajectory for each time t, against the position for each other robots at the same time in their respective paths. The comparison is based on the number of intersections between safety circles with radius r_n. As shown in Fig. 2b two robots with radius r_1 and r_2 are bound to collide having the same distance d_1 and d_2 to travel, at the same speed ($V_1 = V_2$). Fig. 2c depicts a similar situation were the velocities differ enough for the robots not to collide, if $V_1 < V_2$ the second robot arrives first to its final position, followed by the first robot, avoiding collisions.

This GA solves sequentially the movement vectors, then the appropriate velocities to avoid collision. Although the two problems, distance formation and collision avoidance, could be solved using a multi objective GA attempts of doing so exhibit too many mathematical operations and extremely long times, this due to calculating for each robot the collision with other robots for every vector solution of the population.

Fig. 3 presents the two algorithm phases, formation and obstacle avoidance. Given the initial conditions; number of robots N, number of solutions in the population n, crossover probability CP, mutation probability MP, desired distances $D = \{d_{ji}\}$, and aimed precision PD, a random initial population is created, composed of n solutions as shown in Fig. 1, with $180° > \theta_i \geq -179°$ and $0 \text{ mm} > d_i \geq 10{,}000 \text{ mm}$.

This random initial population is subjected to crossover and mutation, Fig. 4 displays the crossover and mutation operator procedure.

For the crossover, two solutions from the initial population are randomly selected and called parents, the crossover probability CP determines the probability percentage of the parents crossing. A first random number between 0 and 1 compared to the CP establish if the parents are to mate and reproduce, for example, if $CP = 0.9$ every random number less or equal than 0.9 will force the parents to mate. If the crossover is positive another random

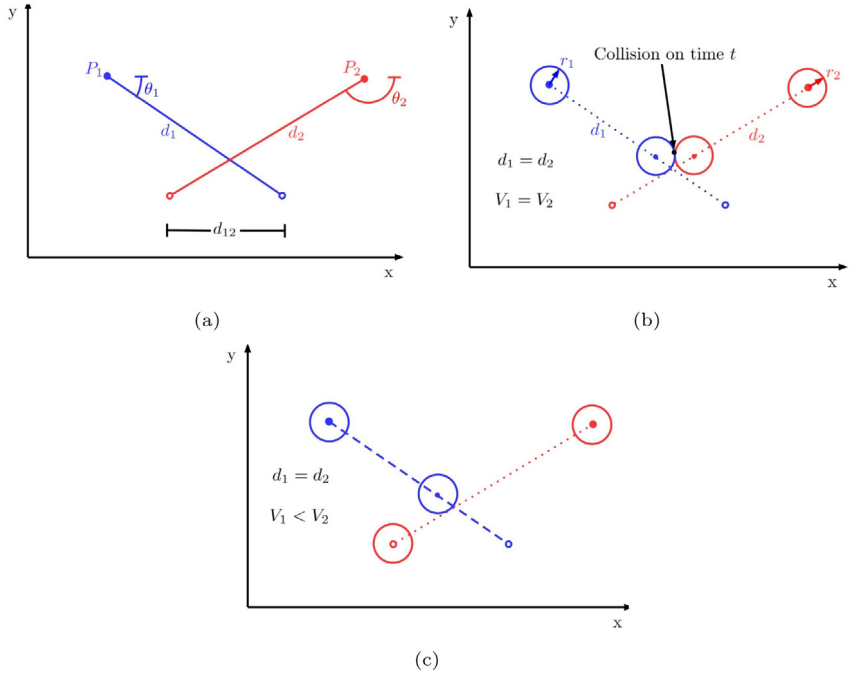


Fig. 2. Formation example with collision.

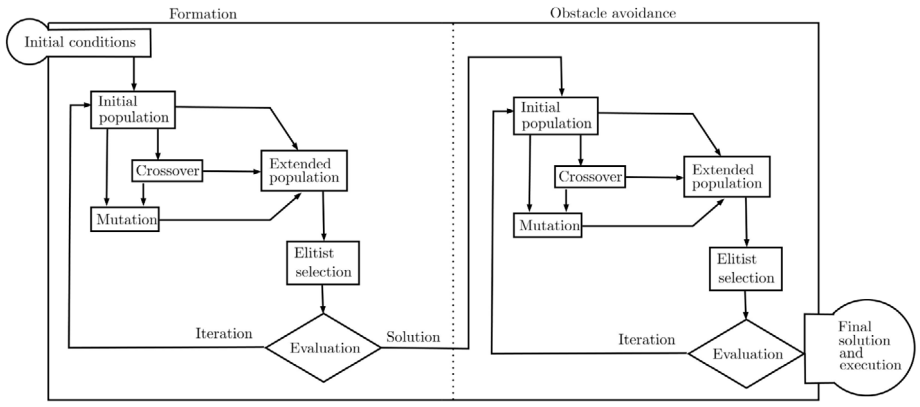


Fig. 3. Proposed GA.

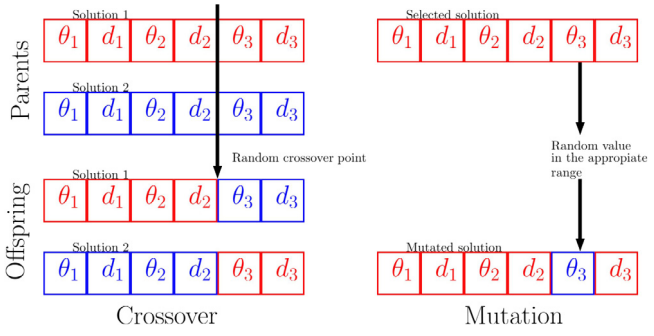


Fig. 4. Crossover and mutation operators.

number selects the crossover point as shown in Fig. 4, then two new solutions, called offspring are created.

The mutation operator evaluates each gene from every solution and, if the mutation probability *MP* falls between the mutation range for example, if $MP = 0.1$ every random number less or equal than 0.1 changes just one gene to a random number inside the appropriate distance or angle range, as presented in Fig. 4.

There are no clear procedures to select *CP* and *MP*, but usually parameter values are mostly selected by conventions [47], *CP* usually selected high, has a high probability to generate diversity, and commonly low probability is assigned to *MP* in order to exit local minima.

The next step is to create a new population, composed by the initial population plus the crossed initial population adding the initial and crossed populations subjected to mutation, this step creates an extended population of $4 \times n$ elements. This population is subjected to an elitist selection, which consist in choosing a new improved population of size n, 50% of the solutions for the new population consist of the best possible solutions, the elite. The rest of the elements for the new population are selected by tournament.

This population is evaluated with the first fitness function, final distance between robots. Each solution is evaluated to determine the Euclidean distances between robots. Then those distances with the desired final distance $D = \{d_{ji}\}$ are used to calculate the error norm. If the error norm of all the solutions is greater than the desired precision, *PD*, the GA restarts using the elite population as the new initial population. Otherwise, GA training is over and obstacle avoidance GA begins.

The obstacle avoidance phase uses the same algorithmic procedure as the formation phase, using the velocity chromosome shown in Fig. 1. The GA operators work in the same way, the evaluation quantifies the number of collisions that occur with each solution. Then the fitness function aims to find a collisions free solution, once such solution is found the algorithm ends. In some cases, when the iterations end, collision free solution cannot be found. In such cases, the distance phase is reinitialized and another solution is found, this is possible thanks to the multiple solutions available for the distance based formation.

4.1. Simulations

Using Matlab® 2017a running on an 64 bits Windows 10® computer, with a Intel® core i7-4500 CPU and 8 GB of RAM, a simulation was made. Considering four robots $N = 4$, being $PD = 80$ mm, $CP = 0.95$, $MP = 0.1$, $n = 1000$, robot safety radius $r = 150$ mm, velocity range $10 \leq V \leq 666$ in mm/s, initial position $\xi_i = [x_i, y_i]$ and desired distance $D = \{d_{ji}\}$ in mm as:

$$\xi = \begin{pmatrix} 5000 & -5000 & -5000 & 5000 \\ 5000 & 5000 & -5000 & -5000 \end{pmatrix} \tag{2}$$

$$D = \begin{pmatrix} 0 & 500 & 500 & 707.1068 \\ 500 & 0 & 707.1068 & 500 \\ 500 & 707.1068 & 0 & 500 \\ 707.1068 & 500 & 500 & 0 \end{pmatrix} \tag{3}$$

Fig. 5a depicts the generated solution vectors and the collision avoidance speed. The obtained chromosome for this solution is

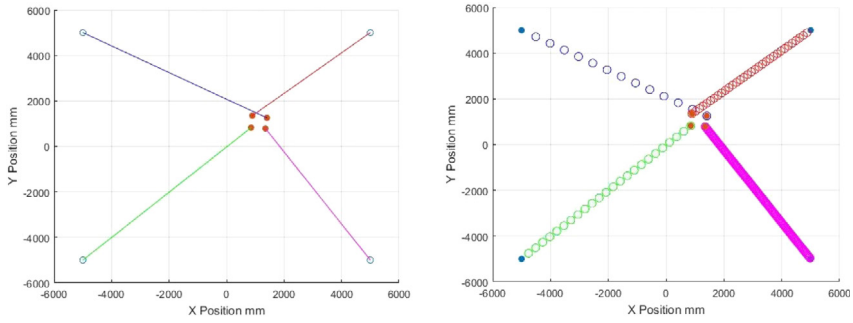
$$[\theta_i, d_i] = \begin{pmatrix} -138.34 & -30.33 & 44.84 & 122.24 \\ 5495.28 & 7421.02 & 8253.64 & 6838.71 \end{pmatrix} \tag{4}$$

this matrix contains four columns that include angle and length that depicts the intended distance and orientation each robot will have to execute to achieve the desired final distances. The second result is the robots velocity which will guarantee no collision between robots

$$V_i = \begin{pmatrix} 166.94 & 565.78 & 343.31 & 38.01 \end{pmatrix} \tag{5}$$

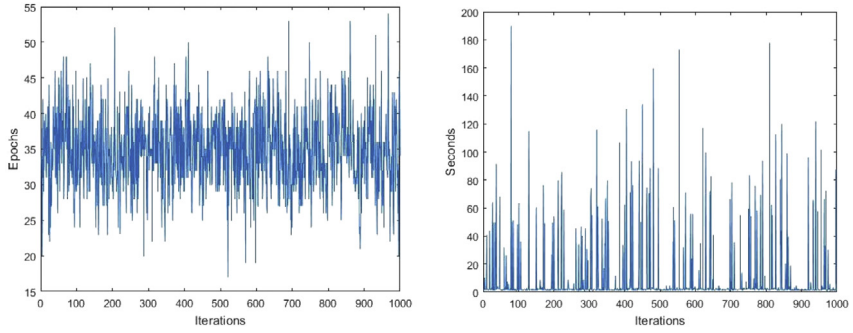
It is easy to note the first and second robot (coloured red and blue respectively) collide. Fig. 5b shows the robot advance, the circles represent the safety radius, the spacing between circles represents the velocity, the more space, the higher velocity. Hence, the blue robot advances faster than the red robot, arriving first to the final position avoiding collision.

Efficiency measures where taken upon one thousand runs of the algorithm, epoch and time results are displayed on Fig. 6. Considering same initial conditions as the last experiment results are shown in Table 1, including maximum and minimum, mean and mode of times and epochs.



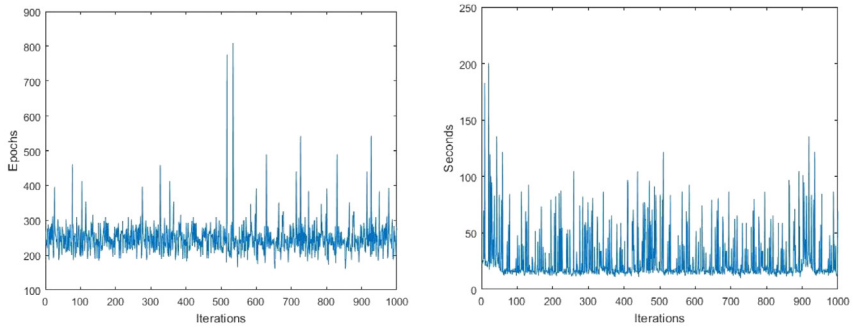
(a) Collision avoidance speed. (b) Distance based formation vectors.

Fig. 5. 4 robots simulation.



(a) Epochs. (b) Time.

Fig. 6. 4 robots simulation statistics.



(a) Epochs. (b) Time.

Fig. 7. 9 robots simulation statistics.

Table 1
Simple GA statistics.

1000 Runs	4 robots simple GA	9 robots simple GA
Max Time	189.92	200.0468
Min Time	0.9075	10.8834
Time Mean	9.3566	26.2794
Time Mode	0.90753	10.8834
Max Epoch	54	809
Min Epoch	17	161
Epoch Mean	34.85	248.392
Epoch Mode	36	229

While the number of epochs and the time overall for each simulations is acceptable, once the algorithm is tested using higher number of robots the performance begins to decline. Fig. 7 shows the results for one thousand runs of the algorithm for $N = 9$ robots, being $PD = 80$ mm, $CP = 0.95$, $MP = 0.1$, $n = 1000$, robot safety radius $r = 150$ mm, velocity range $10 \leq V \leq 666$ in mm/s, initial position $\xi_i = [x_i, y_i]$ and desired distance $D = \{d_{ji}\}$ in mm as Eq. (7) is given in Box I:

$$\xi = \begin{pmatrix} 5000 & 0 & -5000 & -5000 & 5000 & 0 & -5000 & 5000 & 0 \\ 5000 & 5000 & 5000 & 0 & 0 & -5000 & -5000 & -5000 & 0 \end{pmatrix} \tag{6}$$

The analysis presents an increase of time and number of epochs needed to achieve formation with collision avoidance. Statistical results are shown on Table 1. This represents a non proportional raise of almost 3 times the time mean, and near thirteen times the mode with regard to the four robot simulation. The proposed solution is to use parallelism.

$$D = \begin{pmatrix} 0 & 800 & 1600 & 800 & 1131.3 & 1788.8 & 1600 & 1788.8 & 2262.7 \\ 800 & 0 & 800 & 1131.3 & 800 & 1131.3 & 1788.8 & 1600 & 1788.8 \\ 1600 & 800 & 0 & 1788.8 & 1131.3 & 800 & 2262.7 & 1788.8 & 1600 \\ 800 & 1131.3 & 1788.8 & 0 & 800 & 1600 & 800 & 1131.3 & 1788.8 \\ 1131.3 & 800 & 1131.3 & 800 & 0 & 800 & 1131.3 & 800 & 1131.3 \\ 1788.8 & 1131.3 & 800 & 1600 & 800 & 0 & 1788.8 & 1131.3 & 800 \\ 1600 & 1788.8 & 2262.7 & 800 & 1131.3 & 1788.8 & 0 & 800 & 1600 \\ 1788.8 & 1600 & 1788.8 & 1131.3 & 800 & 1131.3 & 800 & 0 & 800 \\ 2262.7 & 1788.8 & 1600 & 1788.8 & 1131.3 & 800 & 1600 & 800 & 0 \end{pmatrix} \quad (7)$$

Box I.

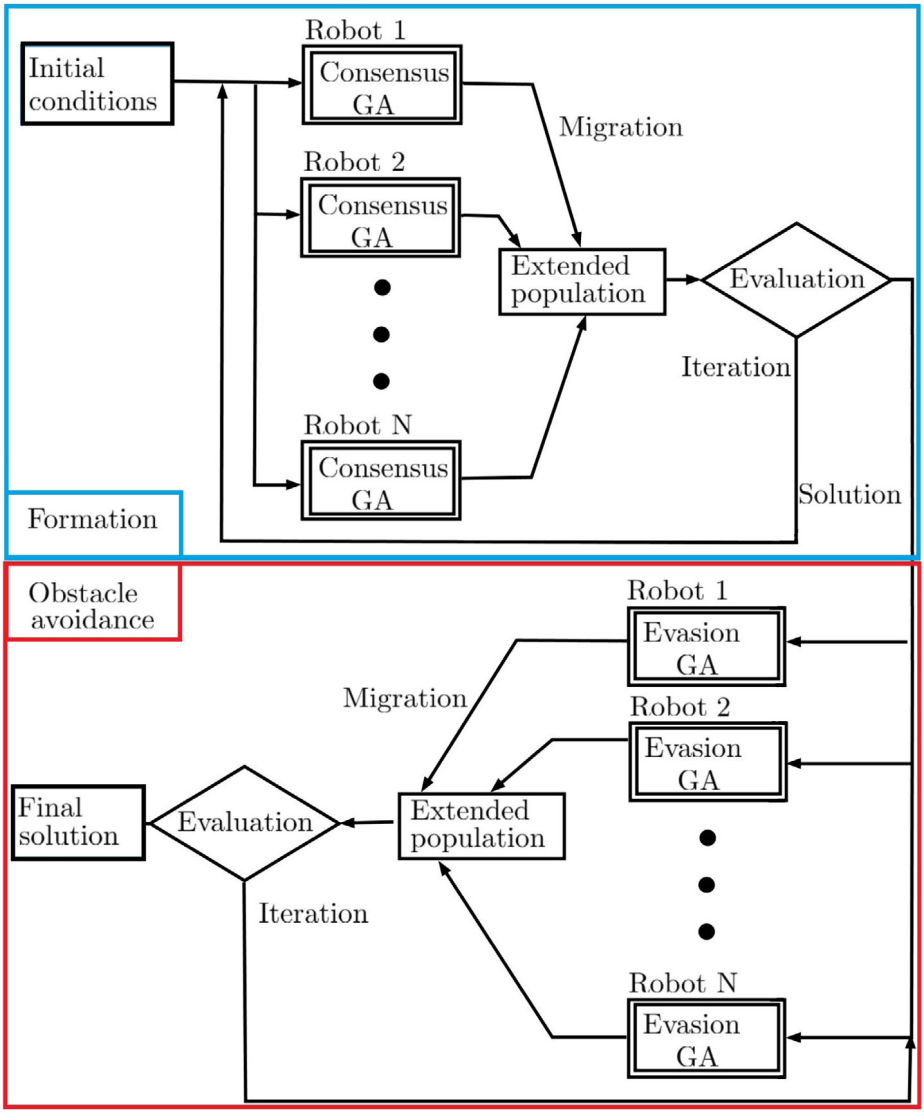
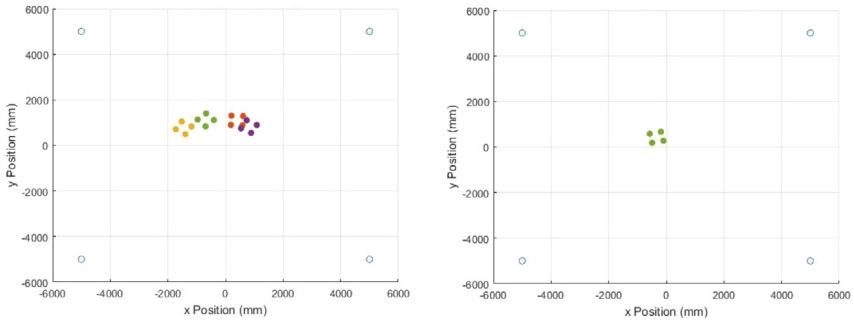


Fig. 8. Parallel GA.

4.2. Parallel GA

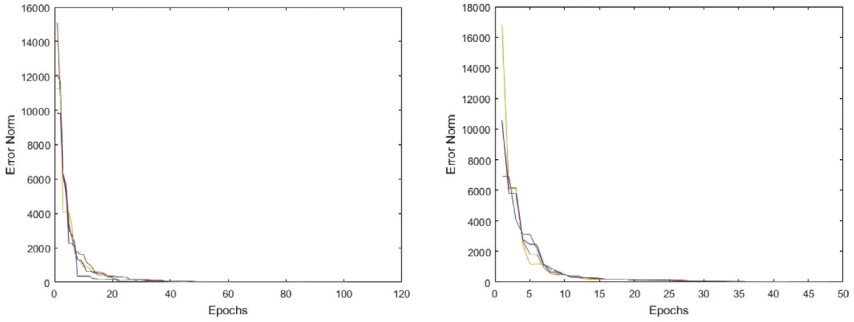
Simulations show that parallel GA improve the algorithm performance. The proposed parallel GA is displayed in Fig. 8. The procedure changes the simple GA running in parallel N algorithms, each one with its own population and operators, adding the migration operator in which the best elements of each population are shared periodically.

Using the same equipment and software shown in Section 4.1, parallelism was simulated, executing sequentially each algorithm until the migration epoch, during migration epoch each of the algorithms would use its own elite population adding all of the others elite population to the selection process. Then the algorithms would restart using the best elements of all algorithms as new population. The performance measurements were carried out using the maximum indicators (maximum time and



(a) No migration. (b) Migration every 3 epochs

Fig. 9. Final position for the parallel GA.



(a) Error without migration. (b) Error with migration every 3 epochs.

Fig. 10. Error norm vs. epochs.

epochs) among the algorithms executed. Matlab parallel computing toolbox was not used because, in the multi core parallelism Matlab automatically generates multiple simultaneous streams of instructions, which, even though it optimizes the algorithm, does not represent the capabilities of a multi robot system, and using a distributed computing system or cluster of several computers was not viable with our research budget.

The migration process generates a distributed intelligence scheme. In order to demonstrate the multi robot distance based formation parallel GA, simulations were conducted using four robots. Considering the robots start at the initial positions $\xi_i = [x_i, y_i]$, all four GAs use it as starting searching points.

Each of these four GAs running in parallel has the same processes for mutation, crossing, and elitism, also the same operator constants (PD, CP, MP) as indicated in Section 4.1. It was chosen in this way to be able to compare the changes integrally when moving to a parallel scheme. Although these variables could change to increase diversity and accelerate the speed of convergence, these changes were relegated to the next phase of development as established in Section 6.

Fig. 9a depicts, the final positions of four GA running in parallel without solutions migration, it can be seen that each GA separately achieves the desired distances, but with very different positions and velocities. The graph in Fig. 9b adds migration every three epochs, the four GA then arrive at the same solution, agreeing not only on the distance and velocity, but also the final solution. Error graphs in Fig. 10a shows the error norm convergence of the parallel GA without migration, it takes more than 60 epochs for all GA's to arrive at their solutions. Contrasting, Fig. 10b, takes all parallel GA's with migration less than 50 epochs to arrive at the common solution.

Statistical results of parallel GA's one thousand simulations with nine robots are shown in Fig. 11. For the parallel GA the maximum time and epoch value is used on every iteration. The performance in epochs and time is compared against the simple GA on Table 2, noting improvement in all aspects.

Table 2
Simple vs. parallel GA.

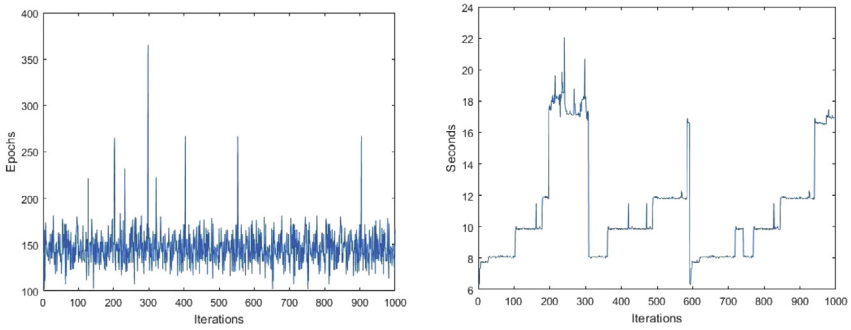
1000 Iterations	9 robots simple GA	9 robots parallel GA
Migration	None	Every 10 epochs
Max Time	200.0468	22.0407
Min Time	10.8834	6.3095
Time Mean	26.2794	11.0488
Time Mode	10.8834	9.9040
Max Epoch	809	365
Min Epoch	161	102
Epoch Mean	248.392	146.408
Epoch Mode	229	151

4.3. Experiments

The approach is tested using the four wheeled mobile robots with mechanum wheels shown in Fig. 12. The posture kinematic model for the omnidirectional robots is described by

$$\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R(\theta_i)} \begin{bmatrix} v_{xi} \\ v_{yi} \\ \omega_i \end{bmatrix} \tag{8}$$

The robots are actuated by Dynamixel servomotors AX − 12 W, and controlled by a microcontroller NXP® model *LPC*1768 with Bluetooth communication to a Windows 7® 64 bits PC computer, with Intel® core *i*7 − 4770 and 16 GB RAM. The position and orientation of the robots were measured by a Vicon® motion capture system composed by 6 cameras model Bonita®. The motion capture measures within an available workspace area of 3 × 7 m. Reflective markers were placed on the top of the



(a) Epochs. (b) Time.

Fig. 11. 9 robots parallel simulation statistics.

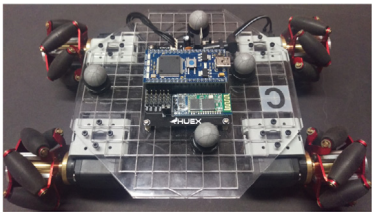


Fig. 12. Omnidirectional wheeled mobile robot.

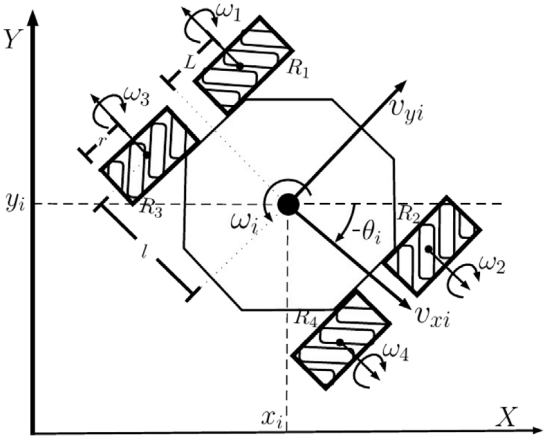


Fig. 13. Omnidirectional wheeled mobile robot diagram.

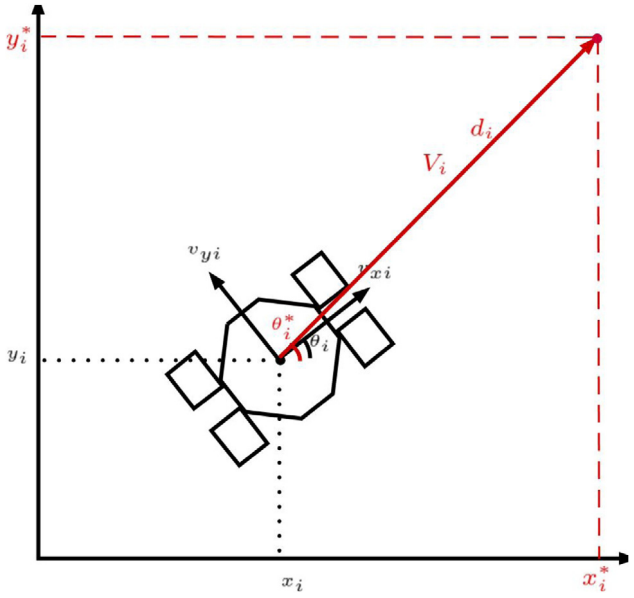


Fig. 14. Robot control diagram.

robots in order to be identified by the Vicon® system. The control algorithm runs at a 117 ms rate with a resolution of ±5 mm.

According to the diagram shown in Fig. 13, the velocities of each wheel can be calculated by

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\\ \omega_4\end{bmatrix} = \frac{1}{r}\begin{bmatrix}1 & 1 & -(L+l)\\ -1 & 1 & (L+l)\\ -1 & 1 & -(L+l)\\ 1 & 1 & (L+l)\end{bmatrix}\begin{bmatrix}v_{x_i}\\ v_{y_i}\\ \omega_i\end{bmatrix}, \quad i = 1,\ldots,N \tag{9}$$

with $V_i = \sqrt{v_{x_i}^2 + v_{y_i}^2}$ and using $L = 10$ cm, $l = 5$ cm and $r = 2.75$ cm.

In order for the robots to achieve the distance, angle and velocity (d_i, θ_i^*, V_i) obtained from the GA and converted to cartesian coordinates (x_i^*, y_i^*), as shown in Fig. 14, the control law uses the input/output linearization given by

$$\begin{bmatrix}v_{xi}\\ v_{yi}\\ \omega_i\end{bmatrix} = R^{-1}(\theta_i)\begin{bmatrix}\overline{fx_i}\\ \overline{fy_i}\\ f\omega_i\end{bmatrix}, \quad i = 1,\ldots,N \tag{10}$$

with

$$\begin{aligned}\overline{fx_i} &= \frac{V_i}{\sqrt{fx_i^2 + fy_i^2 + \epsilon}}\\ \overline{fy_i} &= \frac{V_i}{\sqrt{fx_i^2 + fy_i^2 + \epsilon}}\end{aligned} \tag{11}$$

while $fx_i = -k(x_i - x_i^*)$, $fy_i = -k(y_i - y_i^*)$ and $f\omega_i = -k_\theta(\theta_i - \theta_i^*)$, using $\epsilon = 0.001$, $k = 100$ and $k_\theta = 10$.

The parallel scheme for this experiments was achieved using a master/slave model. The computer is considered the master and the robots the slaves. The master is in charge of the fitness selection over an extended population (sum of all slaves populations) and sending the new elite population to the slaves, then wait for the slaves to finish his processes to receive the new elements of the population from each robot. The slaves are in charge of realizing the crossover, mutation and selection operators on the received population and transmitting the new population. This communication is synchronous, once all robots are ready, the population transmission begins. All the slaves have the same number of elements, and the same operators characteristics (*PD*, *CP*, *MP*).

For the first experiment the experimental set up with four robots $N = 4$ uses $PD = 80$ mm, $CP = 0.95$, $MP = 0.1$, $n = 1000$, robot safety radius $r = 150$ mm, velocity range $10 \leq V \leq 666$ in mm/s, initial position $\xi_i = [x_i, y_i]$ and desired distance $D = \{d_{ji}\}$

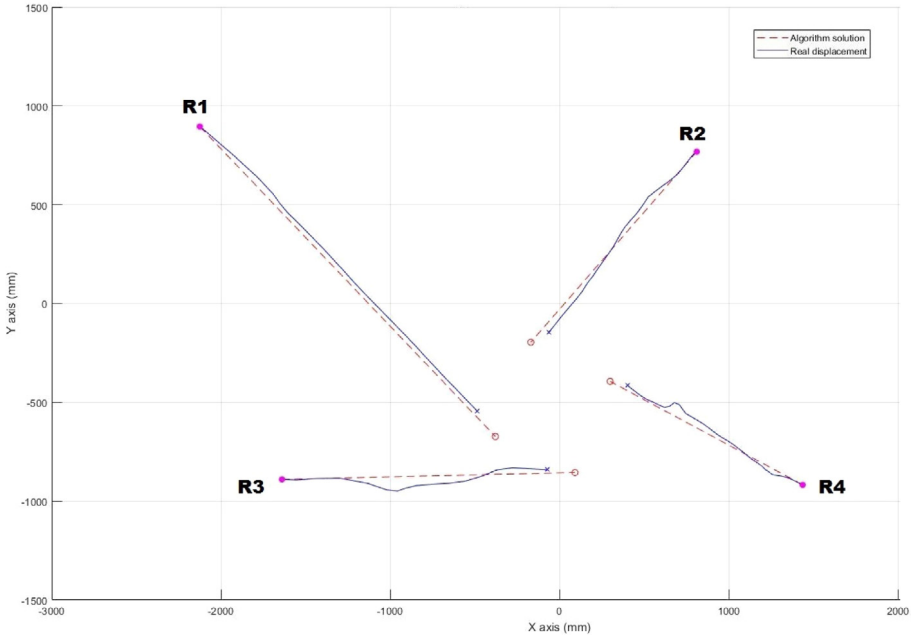


Fig. 15. First experiment robots' trajectory.

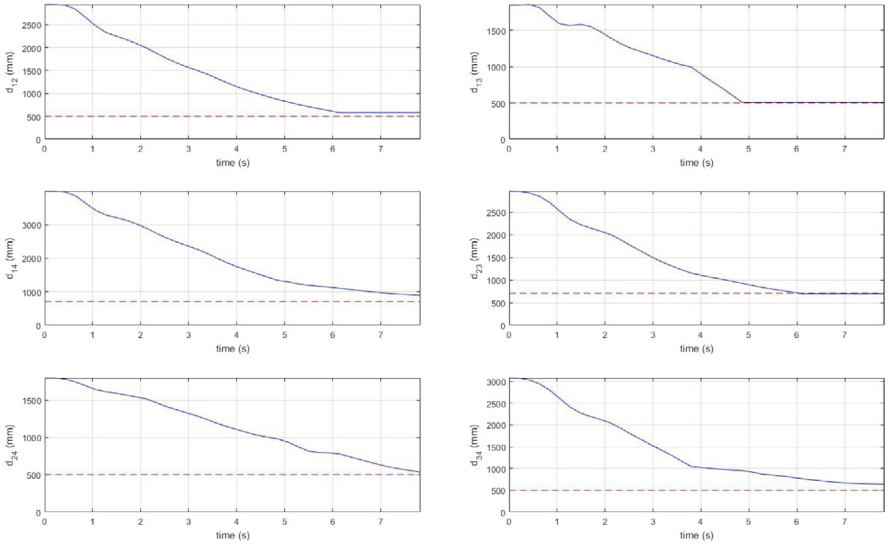


Fig. 16. First experiment robots' final distance.

in mm as:

$$\xi = \begin{pmatrix} -2125.07 & 811.45 & -1639.24 & 1438.03 \\ 895.30 & 768.41 & -890.00 & -917.69 \end{pmatrix} \tag{12}$$

$$D = \begin{pmatrix} 0 & 500 & 500 & 707.1068 \\ 500 & 0 & 707.1068 & 500 \\ 500 & 707.1068 & 0 & 500 \\ 707.1068 & 500 & 500 & 0 \end{pmatrix} \tag{13}$$

The results are presented in Figs. 15, 16 and 17. The algorithm takes 29 epochs to get the final results, the final angle and distance for each robots are:

$$[\theta_i, d_i] = \begin{pmatrix} -41.91 & -135.47 & 1.15 & 155.28 \\ 2347.48 & 1375.13 & 1731.42 & 1253.07 \end{pmatrix} \tag{14}$$

As seen on Fig. 15 the real robots follow the path traced by the solution, and arrive close to the final position.

Fig. 16 shows the final distance evolution between the robots as expected in Eq. (13). The algorithm's calculated velocities are:

$$V_i = \begin{pmatrix} 499.69 & 233.33 & 472.98 & 166.72 \end{pmatrix} \tag{15}$$

Fig. 17 depicts the real robot velocities compared to the ones calculated by the algorithm, the velocities drop to zero once the robot arrives at the final position.

A second experiment uses $PD = 80$ mm, $CP = 0.95$, $MP = 0.1$, $n = 1000$, robot safety radius $r = 150$ mm, velocity range $10 \leq V \leq 666$ in mm/s, being desired distance D same as in Eq. (13) and initial position $\xi_i = [x_i, y_i]$ as:

$$\xi = \begin{pmatrix} -1887.31 & 1444.47 & -2186.78 & 1429.23 \\ 1126.82 & 141.62 & -929.52 & -238.81 \end{pmatrix} \tag{16}$$

The results are presented in Figs. 18, 19 and 20. The algorithm takes 176 epochs to get the final results, the final angle and

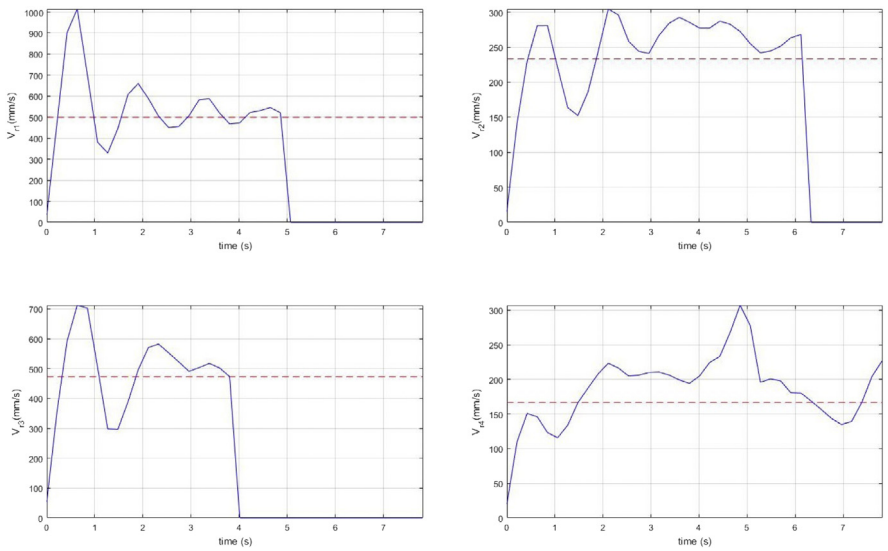


Fig. 17. First experiment robots' velocities.

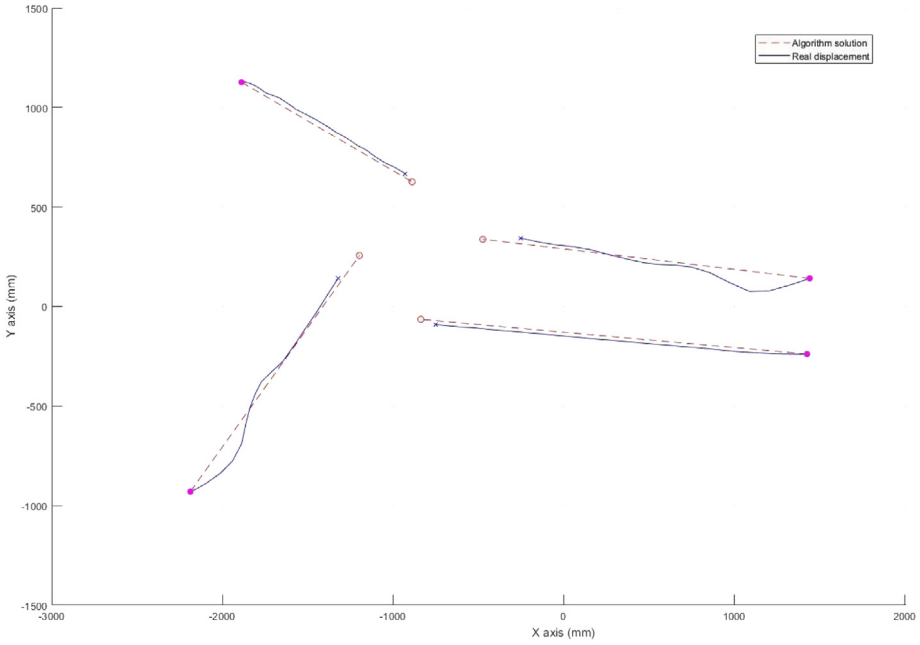


Fig. 18. Second experiment robots' trajectory.

distance for each robots are:

$$[\theta_i, d_i] = \begin{pmatrix} -26.55 & 174.15 & 50.11 & 175.5 \\ 1118.76 & 1927.02 & 1545.33 & 2271.58 \end{pmatrix} \quad (17)$$

Observing Fig. 18 it can be seen that the omnidirectional robots follow the path traced by the GA's solution, and arrive close to the final position.

Fig. 19 shows the final distance evolution between the robots as expected in Eq. (13). The algorithm's calculated velocities are:

$$V_i = \begin{pmatrix} 124.62 & 545.50 & 437.93 & 250.85 \end{pmatrix} \quad (18)$$

Fig. 20 depicts the real robot velocities compared to the ones calculated by the algorithm.

5. Discussion

Simulations results given in Section 4.1 present concept proof of the proposed distance based formation GA and performance upgrade using parallelization. It is feasible to reach consensus in a MRS using DAI through PGAs but more research on performance is needed.

The obstacle avoidance phase is not the best way to achieve collision avoidance. The simulations and experiments used a low number of robots, and while most of the time the obstacle avoidance GA found solutions, sometimes it was difficult or impossible to find solutions, hence the iterative process had to be restarted. Increasing the number of robots will make it harder for the algorithm to find a collision free solution.

The experimental results presented in Section 4.3 were aimed towards the viability of implementation for the distance based formation GA. The results explore the comparison between the

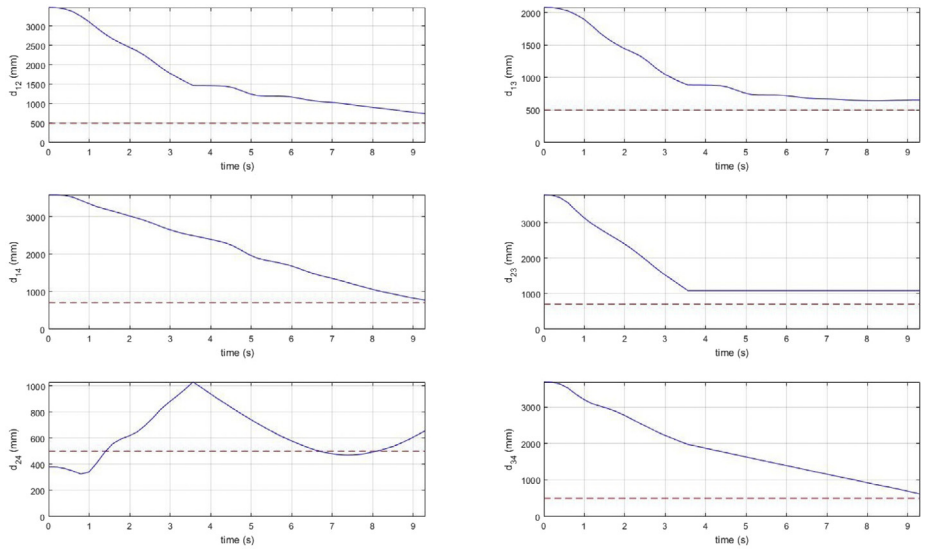


Fig. 19. Second experiment robots' final distance.

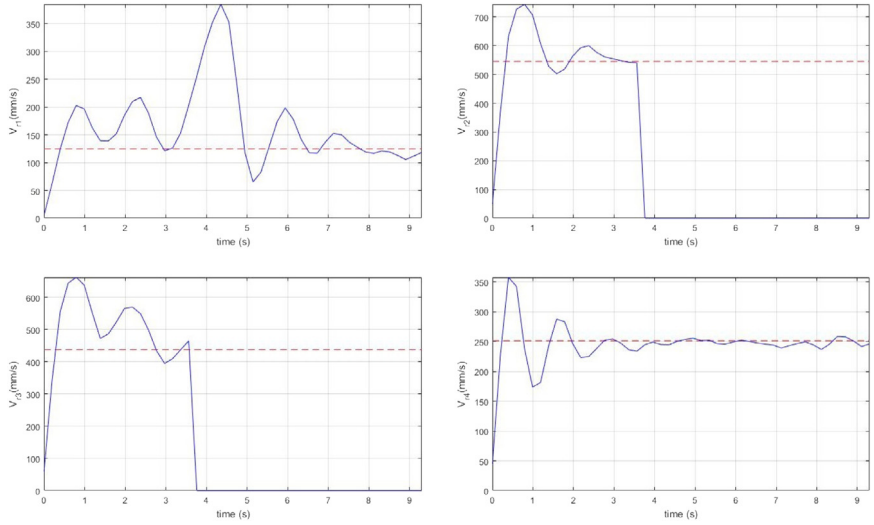


Fig. 20. Second experiment robots' velocities.

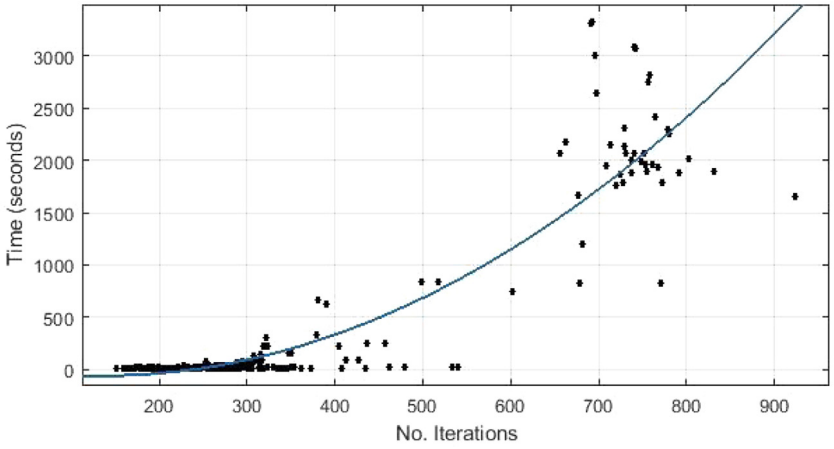
real position, velocity and distance of the real robots versus the simulations. During the experiments some data were obtained on the computation times, however the timing and characteristics of the synchronous transmission by means of Bluetooth, obscured the clear identification of trends on the performance of the parallel algorithm.

Time complexity analysis. An important part of computational complexity theory is time complexity analysis. It is used to describe an algorithm's use of computational resources through the worst case running time, usually expressed using the big Omicron (abbreviated as big-$\mathcal{O}$) notation [48–50]. There is been a lot of work on time complexity analysis for evolutionary algorithms, including genetic algorithms, evolutionary programming and evolutionary strategy [51–56], to mention a few.

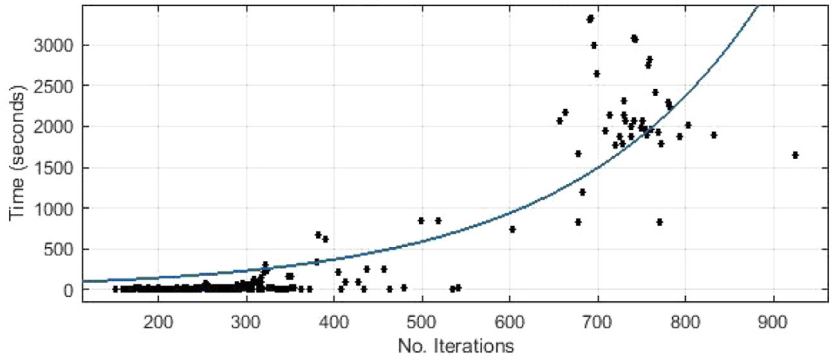
The GA's proposed in this paper, as described on Section 4, both simple and parallel algorithms include premature ending conditions, for the formation phase a lower error norm than that of the desired precision ends the algorithm, and the obstacle avoidance phase ends once a collision free solution is found. This conditions make it difficult to count the algorithm's number of instructions, hence the time complexity analysis was achieved using curve fitting for the time results [53].

The non parallel GA with collision avoidance was executed 1000 times, keeping track of the time and the number of iterations in which the algorithm reaches consensus. The time results versus the number of iterations are shown in Fig. 21. Fig. 21a depicts the fitted curve, a linear second grade polynomial fitting, $f(x) = p_1x^2 + p_2x + p_3$ with coefficients $p_1 = 0.005583$, $p_2 = -1.502$ and $p_3 = 39.68$. Fig. 21b shows an exponential fitted curve $f(x) = ae^{bx}$ with coefficients $a = 58.27$ and $b = 0.004638$. Both fittings were made using all unfiltered data, no outliers detection was made because outlier detection algorithms tend to eliminate most of the points from more than 500 iterations, indicating a linear behaviour.

Table 3 has the goodness fit for both curves, using the sum of squares due to error (SSE), R^2, adjusted R^2, and root mean squared error (RMSE) measurements. It is appropriate to note that the closest SSE and RMSE indicators to 0 belongs to the polynomial fitting, while the nearest R^2 and adjusted R^2 indicators to 1 pertain also to the polynomial fit. This means that the time



(a) Second grade polynomial curve fitting



(b) Exponential curve fitting

Fig. 21. Curve fitting for the non parallel GA.

Table 3
Non parallel GA time curve fitting.

	SSE	R^2	Adjusted R^2	RMSE
Polynomial (Second grade)	2.329e+07	0.8469	0.8454	329.9
Exponential	3.531e+07	0.7679	0.7668	405.3

Table 4
Parallel GA time curve fitting.

	SSE	R^2	Adjusted R^2	RMSE
First grade polynomial	330	0.3328	0.3253	1.937
Second grade polynomial	342.7	0.3072	0.2913	1.985
Third grade polynomial	340.9	0.3108	0.2868	1.991

complexity worst case scenario for this particular GA is $\mathcal{O}(n^n)$ but probably is closer to $\mathcal{O}(n^2)$.

As stated on Section 4.2, the simple algorithm is quite computationally expensive, hence the design of the PGA. The same time complexity analysis through curve fitting was made for the PGA, polynomial fitting results are shown in Fig. 22, from first to third order polynomials. Fig. 22a depicts the first order polynomial, $f(x) = p_1x + p_2$ with coefficients $p_1 = -0.0004414$ and $p_2 = 122.1$. Fig. 22b shows the second grade polynomial fitting, $f(x) = p_1x^2+p_2x+p_3$ with coefficients $p_1 = 7.242e-05$, $p_2 = -0.03153$ and $p_3 = 125.1$. Finally in Fig. 22c the third grade polynomial, $f(x) = p_1x^3 + p_2x^2 + p_3x + p_4$ has coefficients $p_1 = 7.258e-07$, $p_2 = -0.0004304$, $p_3 = 0.07327$, and $p_4 = 118.3$.

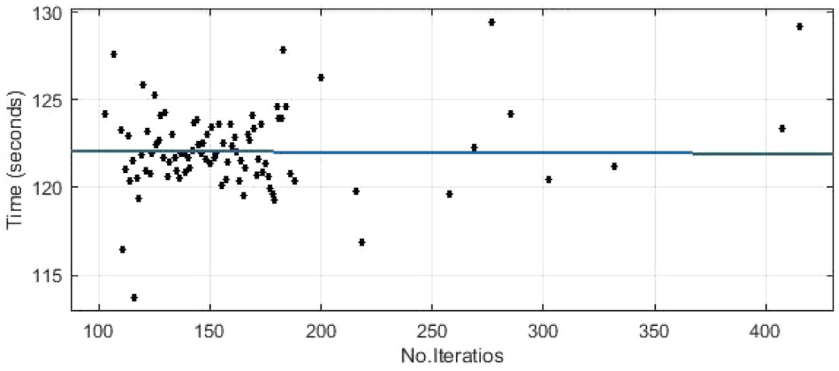
Table 4 presents the goodness results. For this fitting the best results, SSE and RMSE closer to 0, R^2 and adjusted R^2 closer to 1, belongs to the first grade polynomial fitting. The second and third grade fittings are acceptable, this means that the time complexity worst case scenario for this PGA is $\mathcal{O}(n)$. This results confirm the PGA's performance upgrade.

6. Conclusions

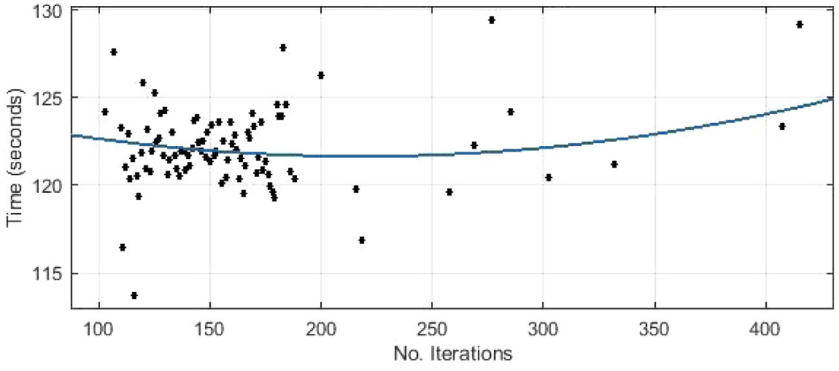
Given the objective of this research presented in Section 3, the proposed algorithm uses GA to achieve distance based formation, with the implementation of two different types of chromosomes, one for the distance solution, and the other for collision avoidance. The algorithm is fast and precise when the number of robots is low, but becomes impractical at higher number of robots. The proposed solution for this is the use of the combined processing power of all robots in a PGA that migrates possible solutions in order to reduce processing time and achieve consensus between the robots to a solution.

As expected simulations results show that the algorithm works, finding solutions to obtain the distance based consensus, while avoiding collisions. One disadvantage of the linear solutions is that the collision avoidance velocity approach not always works, but the flexibility of the distance based consensus allows the algorithm to iterate until a valid solution is achieved.

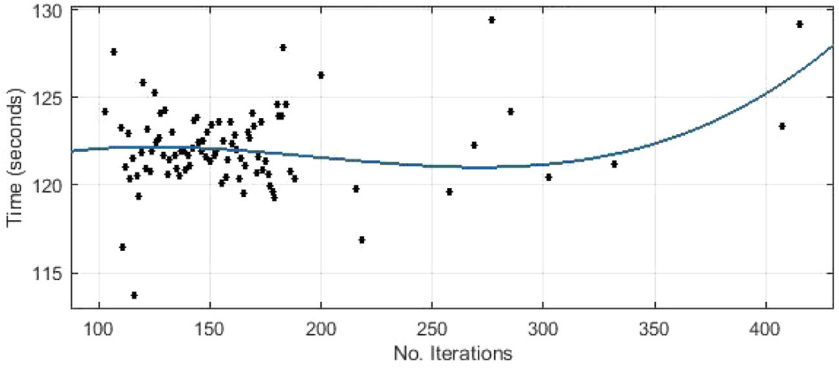
Simulations with a higher number of robots show that the simple GA becomes long, computationally expensive, and prone not to find a valid solution. The parallel solution to this problem



(a) First grade polynomial



(b) Second grade polynomial



(c) Third grade polynomial

Fig. 22. Polynomial curve fitting for the PGA.

demonstrates DAI capable of agreeing on the final distance and velocity solutions, reducing time and computational load.

Experiments show the implementation in real four wheeled omnidirectional robots, demonstrating the practical use of the algorithm to coordinate the movements of MRS.

This algorithm is important because with the growing importance of MRS, finding new and alternative ways to achieve consensus might bring diversity and flexibility in different kinds of situations. The main contribution of this approach is the distributed intelligence point of view aiming towards collective intelligence, in which the robots are not being just coordinated towards a position in plane, they rather agree on a solution using cooperative schemes.

We are focusing our future research on the use of multiobjective evolutionary algorithms to optimize the algorithm. It was not used in this paper because this algorithm is supposed to be a first step on consensus non exact strategies. We are also interested in adding complexity in geometrical shape and velocity curves to the collision free trajectory, as stated in Section 5 the higher the number of robots the harder it will be for the proposed algorithm to find a solution.

The parallel scheme used in this work can be improved using different communication topologies, diverse operators constants and variations on the procedures for the mutation, crossover and elitism. This changes could potentially improve the performance. Additionally we expect to test the algorithms using a distributed computing system or cluster of several computers or microcomputers to better study this parallel algorithms.

The physical MRS platform performance requires much more research than we were able to do in this paper. There is need of a detailed analysis of the hardware implications in performance of the distributed system. One mayor issue is Bluetooth communication and asynchronous communication schemes. We assume that optimizing Bluetooth communication or switching to

a Wi-Fi scheme could increase speed. Particularly, we think that a common genetic pool of solutions from which GAs take and pour data asynchronously could serve to accelerate convergence and reduce the data transmission load.

The use of more robots, and heterogeneous MRS is a future research topic. The heterogeneity could be due to each robot's locomotion (aerial, terrestrial, legged or submarine), and it would be particularly interesting, the diversity between the processing capabilities of different robots.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105929.

References

[1] G. Dudek, M.R.M. Jenkin, E. Milios, D. Wilkes, A taxonomy for multi-agent robotics, Auton. Robots 3 (4) (1996) 375–397, http://dx.doi.org/10.1007/BF00240651.

[2] W. Yu, G. Wen, G. Chen, J. Cao, Distributed Cooperative Control of Multi-agent Systems, John Wiley & Sons, 2017.

[3] Z. Yan, N. Jouandeau, A.A. Cherif, A survey and analysis of multi-robot coordination, Int. J. Adv. Robot. Syst. 10 (12) (2013) 399, http://dx.doi.org/10.5772/57313.

[4] T. Arai, E. Pagello, L.E. Parker, Guest editorial advances in multirobot systems, IEEE Trans. Robot. Autom. 18 (5) (2002) 655–661, http://dx.doi.org/10.1109/TRA.2002.806024.

[5] J. Cortes, M. Egerstedt, Coordinated control of multi-robot systems: A survey, SICE J. Control Meas. Syst. Integr. 10 (6) (2017) 495–503, http://dx.doi.org/10.9746/jcmsi.10.495.

[6] W. Ren, R. Beard, Distributed Consensus in Multi-vehicle Cooperative Control, Springer, London, 2008.

[7] J. Desai, A graph theoretic approach for modeling mobile robot team formations, J. Robot. Syst. 19(11) (2002) 511–525.

[8] Y. Chen, Z. Wang, Formation control: A review and a new consideration, in: International Conference on Intelligent Robots and Systems, 2005, pp. 3181–3186.

[9] K.-K. Oh, M.-C. Park, H.-S. Ahn, A survey of multi-agent formation control, Automatica 53 (2015) 424–440, http://dx.doi.org/10.1016/j.automatica.2014.10.022.

[10] T. Balch, R.C. Arkin, Behavior-based formation control for multirobot teams, IEEE Trans. Robot. Autom. 14 (6) (1998) 926–939, http://dx.doi.org/10.1109/70.736776.

[11] A. Lopez-Gonzalez, E.D. Ferreira, E.G. Hernandez-Martinez, J.J. Flores-Godoy, G. Fernandez-Anaya, P. Paniagua-Contro, Multi-robot formation control using distance and orientation, Adv. Robot. 30 (14) (2016) 901–913, http://dx.doi.org/10.1080/01691864.2016.1159143.

[12] E.D. Ferreira-Vazquez, E.G. Hernandez-Martinez, J.J. Flores-Godoy, G. Fernandez-Anaya, P. Paniagua-Contro, Distance-based formation control using angular information between robots, J. Intell. Robot. Syst. 83 (3–4) (2016) 543–560, http://dx.doi.org/10.1007/s10846-015-0312-1.

[13] E.D. Ferreira-Vazquez, E.G. Hernandez-Martinez, J.J. Flores-Godoy, Fernandez-Anaya, Formation tracking of heterogeneous mobile agents using distance and area constraints, Hindawi, Complexity 2017 (2017) 13, http://dx.doi.org/10.1155/2017/9404193.

[14] E.G. Hernandez-Martinez, J.-J. Flores-Godoy, G. Fernandez-Anaya, A. Lopez-Gonzalez, Formation tracking based on approximate velocities, Int. J. Adv. Robot. Syst. 12 (12) (2015) 181, http://dx.doi.org/10.5772/61944.

[15] S.-M. Lee, H. Myung, Cooperative particle swarm optimization-based predictive controller for multi-robot formation, in: S. Lee, H. Cho, K.-J. Yoon, J. Lee (Eds.), in: Intelligent Autonomous Systems, vol. 12, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 533–541.

[16] E.G. Hernandez-Martinez, J.M.F. Albino, Hybrid architecture of multi-robot systems based on formation control and som neural networks, in: 2011 IEEE International Conference on Control Applications (CCA), 2011, pp. 941–946, http://dx.doi.org/10.1109/CCA.2011.6044358.

[17] Z. Xu, K.S. Choi, Y.G. Kim, J. An, S.G. Lee, An enhanced formation of multi-robot based on a*algorithm for data relay transmission, in: Springer Lecture Notes in Computer Science, vol. 6729, 2011.

[18] C.R.M. Kuppan, M. Singaperumal, T. Nagarajan, Layered framework for formation control of multiple mobile robots - a state based approach, in: 2008 IEEE International Conference on Robotics and Biomimetics, 2009, pp. 2067–2072, http://dx.doi.org/10.1109/ROBIO.2009.4913320.

[19] X.B. Chen, Y. Zhang, Affine transformation of multiple mobile robot formation by generalized ant colony optimization, in: 2008 IEEE Conference on Robotics, Automation and Mechatronics, 2008, pp. 532–536, http://dx.doi.org/10.1109/RAMECH.2008.4681382.

[20] L.E. Parker, Distributed intelligence: overview of the field and its application in multi-robot systems, 2008-03.

[21] G. Weiss, Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT press, 1999.

[22] M. Srinivas, L.M. Patnaik, Genetic algorithms: a survey, Computer 27 (6) (1994) 17–26, http://dx.doi.org/10.1109/2.294849.

[23] E. Alba, J.M. Troya, A survey of parallel distributed genetic algorithms, Complex. 4 (4) (1999) 31–52, http://dx.doi.org/10.1002/(SICI)1099-0526(199903/04)4:4<31::AID-CPLX5>3.3.CO;2-W.

[24] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, first ed., Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[25] D.E. Goldberg, Sizing populations for serial and parallel genetic algorithms, in: Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989, pp. 70–79.

[26] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1992.

[27] S.-M. Chen, C.-Y. Chien, Parallelized genetic ant colony systems for solving the traveling salesman problem, Expert Syst. Appl. 38 (4) (2011) 3873–3883, http://dx.doi.org/10.1016/j.eswa.2010.09.048.

[28] T. Back, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, first ed., IOP Publishing Ltd., Bristol, UK, UK, 1997.

[29] A. Enrique, L. Gabriel, N. Sergio, Parallel metaheuristics: recent advances and new trends, Int. Trans. Oper. Res. 20 (1) (2013) 1–48, http://dx.doi.org/10.1111/j.1475-3995.2012.00862.x.

[30] E. Cantú-Paz, A summary of research on parallel genetic algorithms, 1995.

[31] E. Alba, Parallel Metaheuristics: A New Class of Algorithms, Wiley-Interscience, 2005.

[32] E. Alba, M. Tomassini, Parallelism and evolutionary algorithms, Trans. Evol. Comput. 6 (5) (2002) 443–462, http://dx.doi.org/10.1109/TEVC.2002.800880.

[33] E. Cantú-Paz, D.E. Goldberg, Efficient parallel genetic algorithms: theory and practice, Comput. Methods Appl. Mech. Engrg. 186 (2) (2000) 221–238, http://dx.doi.org/10.1016/S0045-7825(99)00385-0.

[34] L. Kiel, Knowledge Management, Organizational Intelligence and Learning, and Complexity - Volume I, Eolss Publishers Company Limited, 2009.

[35] H. Qu, K. Xing, T. Alexander, An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots, Neurocomputing 120 (Complete) (2013) 509–517, http://dx.doi.org/10.1016/j.neucom.2013.04.020.

[36] C. Lamini, S. Benhlima, A. Elbekri, Genetic algorithm based approach for autonomous mobile robot path planning, Procedia Comput. Sci. 127 (2018) 180–189, http://dx.doi.org/10.1016/j.procs.2018.01.113.

[37] I. Seilonen, K. Koskinen, T. Pirttioja, P. Appelqvist, A. Halme, Reactive and deliberative control and cooperation in multi-agent system based process automation, in: 2005 International Symposium on Computational Intelligence in Robotics and Automation, 2005, pp. 469–474, http://dx.doi.org/10.1109/CIRA.2005.1554321.

[38] A. Alouache, Q. Wu, Tracking control of multiple mobile robot trajectory by genetic algorithms, in: EEA - Electrotehnica, Electronica, Automatica, Vol. 65, 2017, pp. 155–161.

[39] M. Agarwal, N. Agrawal, S. Sharma, L. Vig, N. Kumar, Parallel multi-objective multi-robot coalition formation, Expert Syst. Appl. 42 (21) (2015) 7797–7811, http://dx.doi.org/10.1016/j.eswa.2015.05.032.

[40] C.C. Lin, K.C. Chen, P.Y. Hsiao, W.J. Chuang, W.J. Chuang, Motion planning of swarm robots using potential-based genetic algorithm, 2012.

[41] G. Capi, Z. Mohamed, Multiple robots formation–a multiobjctive evolution approach, Procedia Eng. 41 (2012) 156–162, http://dx.doi.org/10.1016/j.proeng.2012.07.156, International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[42] F. Kobayashi, N. Tomita, F. Kojima, Re-formation of mobile robots using genetic algorithm and reinforcement learning, in: SICE 2003 Annual Conference (IEEE Cat. No.03TH8734), Vol. 3, 2003, pp. 2902–2907.

[43] M. Nazarahari, E. Khanmirza, S. Doostie, Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm, Expert Syst. Appl. 115 (2019) 106–120, http://dx.doi.org/10.1016/j.eswa.2018.08.008.

[44] J.-H. Wei, J.-S. Liu, Mobile robot path planning with eta(3)-splines using spatial-fitness-sharing variable-length genetic algorithm, 2009, pp. 978–983, http://dx.doi.org/10.1109/IROS.2009.5354829.

[45] M. Reza, S. Satapathy, S. Pattnaik, D. Panda, P. Raj, Optimized robot path planning using parallel genetic algorithm based on visible midpoint, Int. J. Eng. Res. Appl. 6 (2016) 14–24.

[46] J. Ferber, Multi-agent system: An introduction to distributed artificial intelligence, J. Artif. Soc. Soc. Simul. 4 (2001).

[47] A.E. Eiben, S.K. Smit, Evolutionary algorithm parameters and methods to tune them, in: Autonomous Search, Springer, 2011, pp. 15–36.
[48] D.E. Knuth, Big omicron and big omega and big theta, SIGACT News 8 (2) (1976) 18–24, http://dx.doi.org/10.1145/1008328.1008329, URL http://doi.acm.org/10.1145/1008328.1008329.
[49] I. Chivers, J. Sleightholme, An introduction to algorithms and the big o notation, in: Introduction to Programming with Fortran, Springer, 2015, pp. 359–364.
[50] S. Bae, Big–o notation, in: JavaScript Data Structures and Algorithms: An Introduction to Understanding and Implementing Core Data Structure and Algorithm Fundamentals, Apress, Berkeley, CA, 2019, pp. 1–11, http://dx.doi.org/10.1007/978-1-4842-3988-9_1.
[51] S. Droste, T. Jansen, I. Wegener, On the analysis of the (1+1) evolutionary algorithm, Theoret. Comput. Sci. 276 (1) (2002) 51–81, http://dx.doi.org/10.1016/S0304-3975(01)00182-7, URL http://www.sciencedirect.com/science/article/pii/S0304397501001827.
[52] C. Witt, Tight bounds on the optimization time of a randomized search heuristic on linear functions, Combin. Probab. Comput. 22 (2) (2013) 294–318, http://dx.doi.org/10.1017/S0963548312000600.
[53] Z. M. Nopiah, M. Ihsan, S. Abdullah, M.N. Baharin, A. Arifin, Time complexity analysis of the genetic algorithm clustering method, 2010, pp. 171–176.
[54] H.-K. Hwang, C. Witt, Sharp bounds on the runtime of the (1+1) ea via drift analysis and analytic combinatorial tools, 2019, arXiv:1906.09047.
[55] B. Doerr, C. Doerr, J. Yang, Optimal parameter choices via precise black-box analysis, in: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16, ACM Press, 2016, http://dx.doi.org/10.1145/2908812.2908950.
[56] Y. Zhou, J. He, A runtime analysis of evolutionary algorithms for constrained optimization problems, IEEE Trans. Evol. Comput. 11 (5) (2007) 608–619, http://dx.doi.org/10.1109/TEVC.2006.888929.