This article can be cited as C. Purcaru, R.-E. Precup, D. Iercan, L.-O. Fedorovici, R.-C. David, F. Drăgan, Optimal Robot Path Planning Using Gravitational Search Algorithm, International Journal of Artificial Intelligence, vol. 10, no. S13, pp. 1-20, 2013.

Optimal Robot Path Planning Using Gravitational Search Algorithm

Constantin Purcaru[1], Radu-Emil Precup[1], Daniel Iercan[1], Lucian-Ovidiu Fedorovici[1], Radu-Codrut David[1], Florin Dragan[1]

[1]Dept. of Automation and Applied Informatics, "Politehnica" University of Timisoara

Bd. V. Parvan 2, 300223 Timisoara, Romania

cpurcaru@gmail.com, radu.precup@aut.upt.ro, daniel.iercan@aut.upt.ro, lucian.fedorovici@aut.upt.ro, davidradu@gmail.com, florin.dragan@aut.upt.ro

ABSTRACT

This paper proposes a new Gravitational Search Algorithm (GSA)-based approach for generating an optimal path for a robot travelling in partially unknown environments in the presence of multiple (static or dynamic) obstacles. The GSA-based approach is expressed as an algorithm which computes an optimal path for a robot that travels from an initial point to a target point while avoiding all the known obstacles in the environment but also any other static or dynamic object that could appear in the path of the robot to the target point. To validate the new approach for the path planning, the new algorithm is employed in the generation of obstacle-free paths for different robots that are participating in different missions in the framework of the nRobotic platform developed at the "Politehnica" University of Timisoara, Romania. A comparison focused on the resulted path length and performance with another well-known evolutionary algorithm represented by the Particle Swarm Optimization used for path planning is performed.

Keywords: Gravitational Search Algorithm, optimal robot path planning, Particle Swarm Optimization.

Mathematics Subject Classification: 82C21, 93A30

Computing Classification System: I.2.3, I.2.9

1. INTRODUCTION

When designing and implementing a platform for interconnecting different types of mobile robots one of the most important tasks is the path planning that generates a collision-free trajectory for each of the robot that takes part at a specific mission (Purcaru et al., 2012). The literature contains several research papers that propose different path planning algorithms using classical approaches or evolutionary algorithms in terms of on-line and off-line path planning algorithms. An overview of the research progress in this field is presented in (Raja and Pugazhenthi, 2012).

The classical approaches to path planning algorithms include the cell decomposition (Lozano-Perez and Wesley, 1979; Kallem et al., 2012), the potential field method (Khatib, 1986; Pozna et al., 2009), the vector field histogram (Borenstein and Koren, 1991; Selekwa et al., 2008), etc. These approaches give good results, but the main disadvantages are the increased computation time and the possibility that the robot could get stuck in a cyclic behavior in some dead-end situations; it is possible to define a set of rules to resolve the cyclic behavior, but the resulting path still would be a non-optimal one as mentioned in (Borenstein and Koren, 1991).

The path planning problem for mobile robots is a non-deterministic polynomial time hard (NP-hard) problem. In the recent years the evolutionary algorithms are becoming widely employed in solving this problem with improved performance versus the classical approaches.

Some of the evolutionary algorithms used for generating optimal trajectories for mobile robots are Ant Colony Optimization (ACO) (Fan et al., 2003; Tan et al., 2007; Garcia et al., 2009; Brand et al., 2010; Chia et al., 2010), genetic algorithms (Gemeinder and Gerke, 2003; Tu and Yang, 2003; Hu and Yang, 2004; Tuncer and Yildirim, 2012), Particle Swarm Optimization (PSO) (Qin et al., 2004; Chen and Li, 2006; Saska et al., 2006; Wang et al., 2006; Masehian and Sedighizadeh, 2010), migration algorithms (Vaščák and Paľa, 2012), etc.

The evolutionary algorithms are also used in combination with classical algorithms. An ACO algorithm is proposed in (Mei et al., 2006) as a global planner while the local planner uses an Artificial Potential Field algorithm. The potential field and the motion dynamics model are combined in (Park and Kim, 2008) to define a PSO algorithm. The PSO is used as the general planner and the Probabilistic Roadmap method is used as a local planner in (Masehian and Sedighizadeh, 2010).

This paper proposes a new approach to the generation of optimal collision-free trajectories in partially-known environments with multiple static or dynamic objects which incorporates a Gravitational Search Algorithm (GSA). A new algorithm is formulated and compared with another algorithm which incorporates a PSO algorithm. The presentation is focused on the implementation of the algorithms in the framework of the nRobotic platform developed at the “Politehnica” University of Timisoara, Romania. Our approach is important because it can be applied in the optimal path planning in combination with other optimization algorithms (Angelov et al., 2008; Klančar et al., 2011; Kovács et al., 2011; Lughofer et al., 2011; Bouhmala, 2012; Farahani et al., 2012) which can solve optimization problems in various applications (Preitl and Precup, 1997; Precup et al., 2004; Hermann et al., 2009;

Haber et al., 2010; Johanyák, 2010; Wilamowski and Yu, 2010; Linda and Manic, 2011; Milojković et al., 2010; Chiang and Roy, 2012; Kayacan et al., 2012).

The paper is organized as follows. The main aspects concerning the implementation of our GSA algorithm are discussed in Section 2. Experimental results with different types of objects in known or partially known environments and a comparison of the proposed algorithm with a PSO one are presented in Section 3. The concluding remarks are highlighted in Section 4.

2. GSA-BASED OPTIMAL PATH PLANNING APPROACH

2.1 Gravitational Search Algorithm

The GSA is an optimization algorithm based on the law of gravity (Rashedi and Nezamabadi, 2009), the performance of the agents being measured by their masses with the gravitation force playing a role of direct form of communication between the agents A different version of the gravitation law is given in (Gauci et al., 2012). This subsection will use a part of the implementation details reported in (Precup et al., 2013).

This paper adapts the GSA to be used in generating an optimal trajectory from an initial point towards a target point while avoiding all the obstacles in the environment. The environment is characterized by known obstacles or obstacles discovered with the ultrasonic or infrared sensors mounted on the robots.

GSA uses a population of agents and the optimization results from the movement of these agents in the search direction. The algorithm starts by randomly placing the agents in the search space, the performance of each agent being measured by its mass, a heavier mass represents a more efficient agent with a higher attraction. The (inertial) mass of the agent i, at the time t which also represents the iteration index, is calculated using the following equations:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}, \tag{1}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)}, \tag{2}$$

where N is the total numbers of agents in the search space, $i = 1...N$, $fit_i(t)$ represents the value for the fitness function at the time t, while $worst(t)$ and $best(t)$ are the worst and the best evaluation of the fitness function; for a minimization problem the values the values are calculated using:

$$worst(t) = \max_{i=1,N} fit_i(t), \tag{3}$$

$$best(t) = \min_{i=1,N} fit_i(t). \tag{4}$$

The position of an agent in the search space is defined by the following vector position:

$$\mathbf{X}_i(t) = (X_i^1(t),..., X_i^d(t),..., X_i^q(t)),\ i = 1...N\ , \tag{5}$$

where X_i^d is the position of the agent i in the dimension $d = 1...q$ and q is the dimension of the search space.

An agent moves in the search space at the time t and in the dimension d with a velocity v_i^d updated using the formula:

$$v_i^d(t) = \sigma \cdot v_i^d(t-1) + a_i^d(t-1), \tag{6}$$

where σ is a random number in the $[0,1]$ interval and a_i^d is the acceleration of the agent given by the equation:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)}, \tag{7}$$

where F_i^d is the total force that acts on the agent in the dimension d, which is equal with the sum of the other agents forces exercised on the agent:

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} \sigma \cdot F_{i,j}^d(t)\,, \tag{8}$$

$$F_{i,j}^d(t) = G(t) \cdot \frac{M_{p,i}(t) M_{a,j}(t)}{E_{i,j}(t) + \varepsilon} [X_i^d(t) - X_j^d(t)], \tag{9}$$

where ε is a small constant, $G(t)$ represents the gravitational constant at time t computed in terms of (Precup et al., 2013), $M_{p,i}(t)$ is the passive gravitational mass of the agent j and $M_{a,j}(t)$ is the active gravitational mass of the agent j, while $E_{i,j}(t)$ represents the Euclidian distance between the agents i and j:

$$E_{i,j}(t) = \sqrt{\sum_{d=1}^{q} [X_j^d(t) - X_i^d(t)]^2}\ . \tag{10}$$

After the acceleration and the velocity are updated the position of the agent in the search space can be calculated with the following formula:

$$X_i^d(t) = X_i^d(t-1) + v_i^d(t)\,. \tag{11}$$

2.2 GSA-based off-line path planning algorithm

Since this paper focuses on a new algorithm that generates a collision-free trajectory for a robot in a partially known environment, the first step of this new algorithm is to generate a global path for the robot to follow (off-line path planning) using only the available information about the environment in which the robot is moving, namely the positions of some of the objects and the position of the other robots. After the global path is generated the robot will follow the resulted trajectory while continuously reading data from the environment using both ultrasonic and infrared sensors. If the robot discovers an object that blocks the global path, the robot will stop and update the map with the new information (the exact location of the newly discovered object) and it will recalculate a new global path (this time from the new position of the robot).

This subsection presents the way in which the global path is obtained, the on-line path planning algorithm being detailed in the next subsection.

The objective of the algorithm is to create an optimal path from an initial point to a target point in the $q = 2$ -dimensional search space which is exactly the solution space of the mobile robot. The objective is thus to minimize the length of the path that the robot needs to travel in order to reach safely (i.e., in the conditions of obstacle avoidance) the target point. With this respect the agent position and velocity actually represent the mobile robot's position and velocity. The objective function that needs to be minimized in order to obtain a shorter path with the help of the GSA described in the previous subsection is the Euclidian distance between the target point and each agent position:

$$E_i(t) = \sqrt{(X_f - X_i(t))^2 + (Y_f - Y_i(t))^2},\ i = 1...N, \tag{12}$$

where (X_f, Y_f) is the target point and $\mathbf{X}_i(t) = (X_i(t), Y_i(t))$ is the agent vector position which represents the current position of the agent in the search space at the time (iteration) t.

The relation between the objective function E_i and the fitness function of an agent at the time t is:

$$fit_i(t) = E_i(t),\ i = 1...N. \tag{13}$$

The flowchart for the proposed algorithm is presented in Figure 1.

The step 1 of the algorithm generates a collection of agents, each agent having a random velocity value in the [-0.2, 0.2] interval. The standard GSA algorithm randomly initializes the positions of the agents in the search space (Rashedi and Nezamabadi, 2009), but this approach is not suitable for the problem we are trying to solve, since our algorithm needs to find the trajectory path starting from an initial known point. Other authors resolved this problem by initially placing the particles used in the PSO algorithm around the initial position of the robot sensing area, thus the number of created particles depends on the number of sensors mounted on the robot (Masehian and Sedighizadeh, 2010). In our implementation of the GSA, the initial position of all created agents is exactly the same location, namely the initial position of the robot, in order to eliminate the dependency between the number of agents and the number of sensors on the robot. Using this approach the algorithm is able

to create as many as agents are required in order to obtain more feasible trajectory paths to the target point.

After the population of agents the algorithm repeats a number of steps, until all the agents reach the target point or a maximum number of iterations is reached. During each iteration the algorithm needs to determine the best and the worst agent from the iteration. This is done in the step 2 of the algorithm by calculating the fitness function for each agent, namely by calculating the Euclidian distance between the current position of the agent and the final position to which the robot needs to travel.

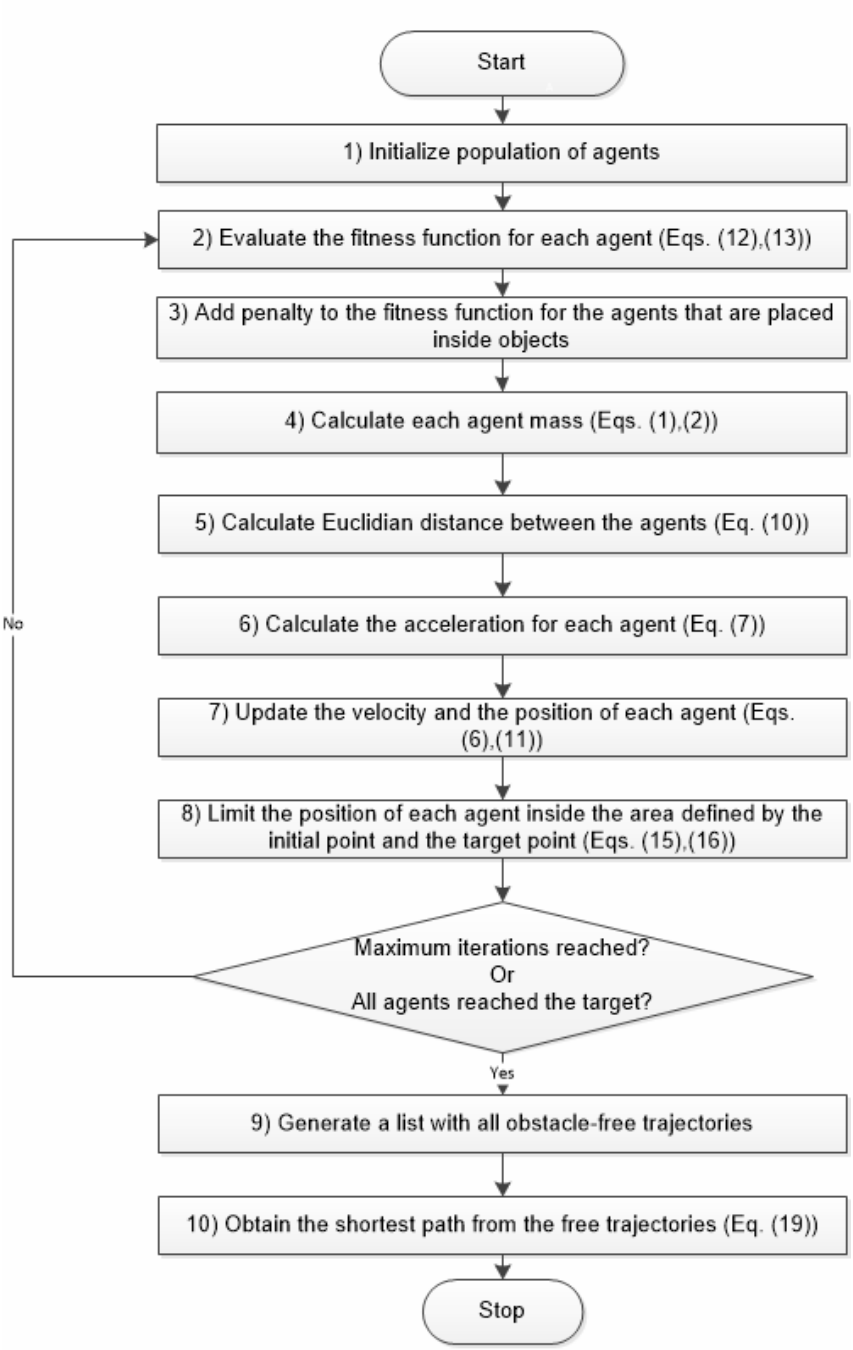

Figure 1. Flowchart of the GSA-based off-line algorithm.

The best agent per iteration is the agent with the lowest Euclidian distance. It is possible that the agent with the best position to be inside of an obstacle, therefore it is necessary to add a penalty to the value of the agent fitness, decreasing the agent performance and force of attraction for the other agents. It is assumed this way that the other agents do not follow an agent with a trajectory that contains points inside obstacles or that the line that connects two successive points of the generated trajectory does not intersect with any known obstacle. The augmented value of the fitness function is $afit_i(t)$:

$$afit_i(t) = fit_i(t) + \alpha,\ i = 1...N, \tag{14}$$

where $\alpha > 0$ is a constant which represents the value of the penalty that will be applied to the agent fitness. A more detailed view of step 3 that consists in adding penalties to an agent performance is given by the flowchart presented in Figure 2. Therefore $afit_i(t)$ is used in the algorithm instead of $fit_i(t)$ after step 3, but we will use as follows the same notation for these two fitness functions for the sake of simplicity.

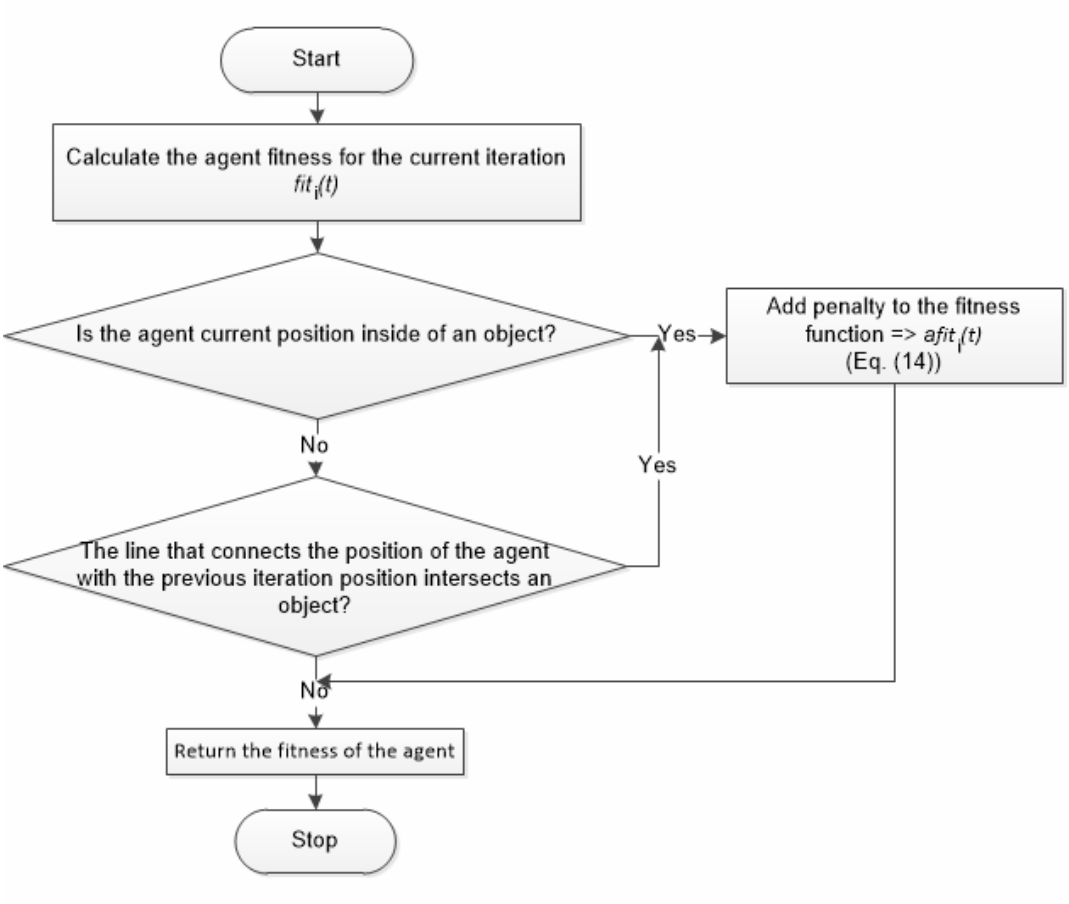

Figure 2. Flowchart for adding penalties to agent fitness.

Steps 5 and 6 of the algorithm consist in calculating the agent masses and the Euclidian distances between all the created agents. In the step 7 from the Figure 1 the new velocity and the new position for the particle can be calculated using equations (7) and (11).

A set of constraints to the agent next position is used in step 8 of the algorithm, namely when updating the next position for each agent an important issue is to keep the resulted position in the search space area characterized by the lower and upper limits of $X_i(t)$ and $Y_i(t)$ coordinates, represented by $X_{\min}$ and $Y_{\min}$, and $X_{\max}$ and $Y_{\max}$, respectively. The lower limit and the upper limit are calculated on the basis of the initial point and the target point and the constraints, which guarantee $\overline{X}_i(t) \in [X_{\min}, X_{\max}]$ and $\overline{Y}_i(t) \in [Y_{\min}, Y_{\max}]$, are expressed as:

$$\overline{X}_i(t) = \begin{cases} X_{\min} & \text{if } X_i(t) < X_{\min}, \\ X_i(t) & \text{if } X_i(t) \in (X_{\min}, X_{\max}), \\ X_{\max} & \text{if } X_i(t) > X_{\max}, \end{cases} \tag{15}$$

$$\overline{Y}_i(t) = \begin{cases} Y_{\min} & \text{if } Y_i(t) < Y_{\min}, \\ Y_i(t) & \text{if } Y_i(t) \in (Y_{\min}, Y_{\max}), \\ Y_{\max} & \text{if } Y_i(t) > Y_{\max}, \end{cases} \tag{16}$$

where the constrained values of the $X_i(t)$ and $Y_i(t)$ coordinates are $\overline{X}_i(t)$ and $\overline{Y}_i(t)$, respectively. The constrained agent position vector $\overline{\mathbf{X}}_i(t) = (\overline{X}_i(t), \overline{Y}_i(t))$ is used in the algorithm instead of the agent position vector $\mathbf{X}_i(t) = (X_i(t), Y_i(t))$ after step 8. We will use as follows the same notation for these two vectors for the sake of simplicity.

In order to retain all points through which an agent passes while moving through the search space, a new property is added to the agent. This property concerns the trajectory points sets $T_{i,t}$ that will be updated at each iteration of the algorithm, adding to it the agent position at each iteration:

$$T_{i,t} = \{(X_i(0), Y_i(0)), (X_1(1), Y_1(1)), ..., (X_i(t), Y_i(t))\}, \; i = 1...N. \tag{17}$$

where $\mathbf{X}_i(0) = (X_i(0), Y_i(0))$ is the initial position of the agent i.

After all agents reached the target point or the maximum number of iteration is reached, each agent will have the list of trajectory points updated .The trajectory path for each agent will be constructed by connecting with lines all successive points of the $T_{i,t_{\max}(i)}$ sets, where $t_{\max}(i)$ represents the maximum number of iterations for which the algorithm runs for the agent i, $t = 0...t_{\max}(i)$.

It is possible that some of the agent trajectory points to be placed inside objects or the line connecting two successive points of the agent trajectory to intersect obstacles. Therefore in step 9 of the

algorithm it is necessary to eliminate all agent trajectories that are not collision-free, that means the trajectory points sets $T_{i,t_{\max}(i)}$, $i \in S_{cf} \subset \{1,2,...,N\}$, where S_{cf} is the set of agents which lead to collision-free trajectories.

After the S_{cf} set is generated, which actually corresponds to a list of collision-free, the shortest path from this list must be determined. In the last step of the algorithm, this path is obtained by the calculation of the total Euclidian distance for each trajectory:

$$D_i = \sum_{t=1}^{t_{\max}(i)} \sqrt{[X_i(t) - X_i(t-1)]^2 + [Y_i(t) - Y_i(t-1)]^2}, \; i \in S_{cf} . \tag{18}$$

With this respect, the optimization problem, that gives the index $\hat{i}$ of the agent which corresponds to the shortest path, is expressed as:

$$\hat{i} = \arg\min_{i \in S_{cf}} D_i . \tag{19}$$

The agent $\hat{i}$ is next used, by means of (19), to compute the trajectory points set $T_{\hat{i},t_{ma}(\hat{i})}$ which corresponds to the shortest path.

2.3 On-line path planning algorithm

The GSA-based algorithm presented in Subsection 2.2 generates an optimal collision-free trajectory for the robot to follow to the target point by using only the information available about the environment, therefore assuming that all objects positions from the environment are known in advance. In many scenarios this information about the environment in which the robot needs to travel is not available entirely or it is not available at all, therefore an on-line path planning algorithm needs to be used in combination with the GSA-based one.

Using this on-line algorithm the robot can get to the target point while avoiding all known obstacles in advance but also any new object that might appear in its way. A similar on-line algorithm which uses the piecewise Hermite spline in order to generate a smooth path is discussed in (Boonporm, 2011).

The proposed algorithm will be used in different missions on the nRobotic platform developed at the “Politehnica” University of Timisoara, Romania, some missions requiring at least two robots in order to successfully start as shown in (Purcaru et al., 2012). Therefore is needed for the on-line path planning algorithm to take into consideration the position of the other robots in the environment (which is known at every point) and avoid any collision between them.

Our algorithm can be used even in completely unknown environments, by firstly generating an optimal global path with the help of the GSA-based off-line algorithm. The robot will follow the generated path while continuously reading data with the help of the ultrasonic and/or infrared sensors. When an

obstacle that blocks the global path is detected the robot needs to generate a new path to the target point.

The flowchart of the on-line path algorithm is presented in Figure 3.

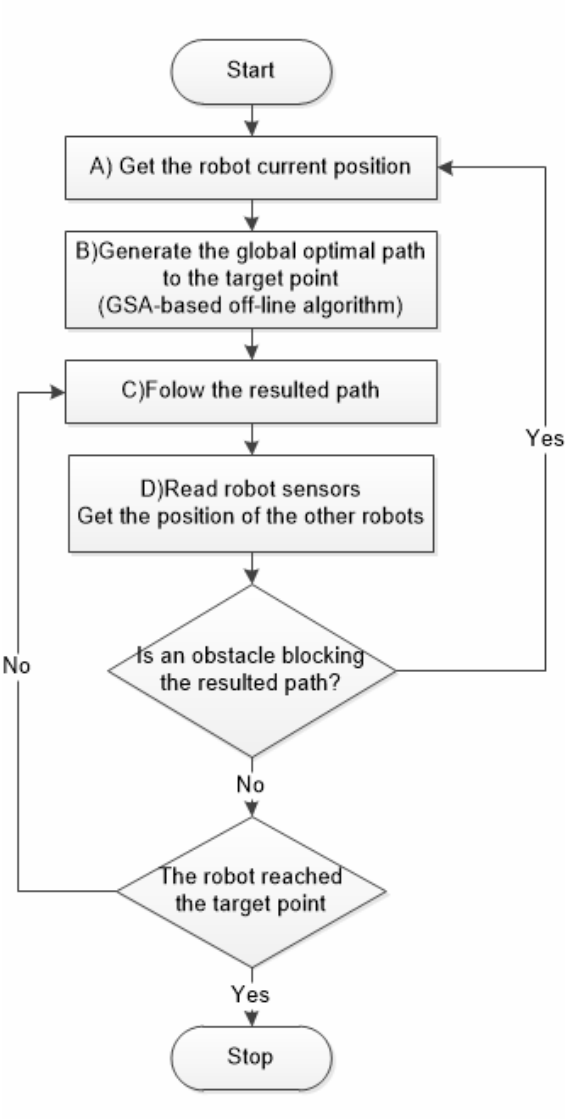

Figure 3. Flowchart of on-line path planning algorithm.

The position of the robot is used at step A of the algorithm as the initial point from which the GSA-based off-line algorithm finds the optimal collision-free trajectory path to the target point. Based on the environment map (with the known obstacles), step B generates the optimal off-line trajectory (19) that will be followed by the robot in order to arrive at the target point.

Once the global trajectory is obtained the robot starts to move in the environment, following the trajectory points from the shortest path generated with the algorithm given in Subsection 2.2.

The robot continuously reads information with the help of the infrared and ultrasonic sensors, while also using information from the nRobotic platform about the position of the other robots that participate at the mission. Using this information the robot is able to determine if there is an obstacle or another robot that blocks the path generated at step B. If the path is clear the robot will continue its movement until the target point is reached.

If there is an obstacle that blocks the path the robot the robot will stop and update the map of the environment with the new values. After the map is updated with the new information about the location of the newly discovered object the algorithm will return to step A; a new path to the target point is generated, this time not from the initial point but from the new position of the robot in the environment.

3. EXPERIMENTAL RESULTS

In order to test the proposed GSA-based path planning algorithm several simulations were run on the nRobotic platform. The simulations were run in completely known environments to properly test the GSA-based offline path planning algorithm but also in partially-known environments where the on-line algorithm is used for properly detecting any new obstacle that may appear in the path of the robot to the target point.

3.1 GSA-based off-line algorithm

A first case which deals with a simple scenario is illustrated in Figure 4, where different trajectories are generated for the same environment with two known objects.

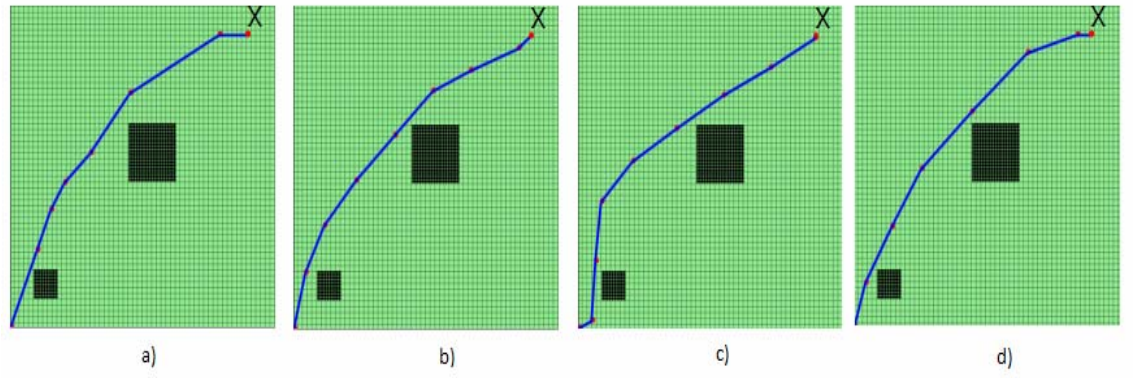

Figure 4. GSA-based solutions in a simple environment.

Several trajectories generated with a PSO algorithm are presented in Figure 5. A more complex environment with objects of different types and the generated trajectories by the proposed GSA-based path planning algorithm are shown in Figure 6.

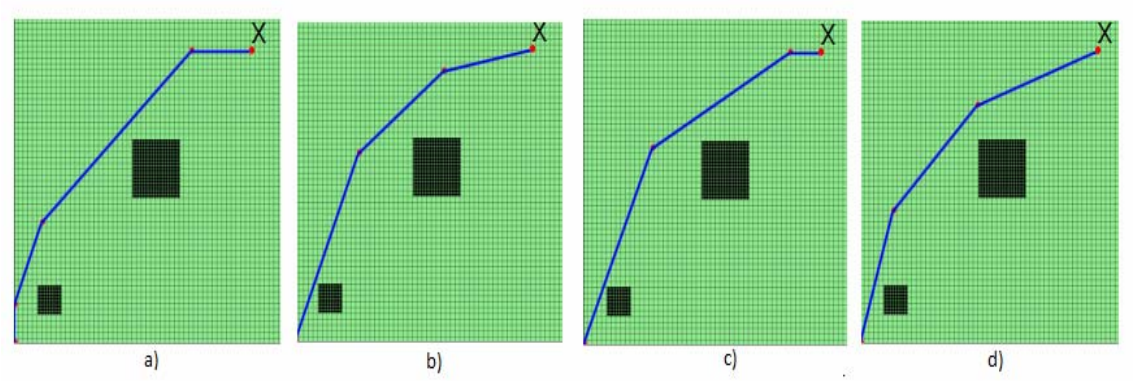

Figure 5. PSO-based solutions in a simple environment.

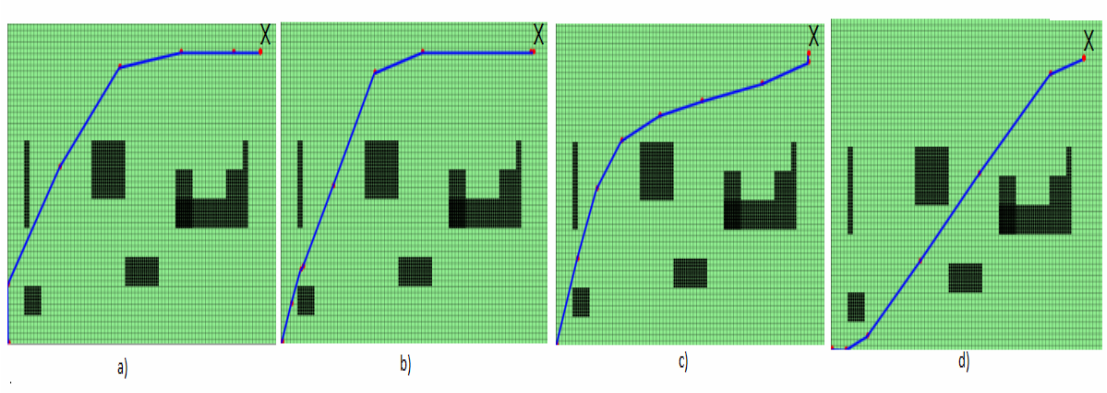

Figure 6. GSA-based solutions in a complex environment.

The solutions generated by the PSO algorithm for the same environment are presented in Figure 7.

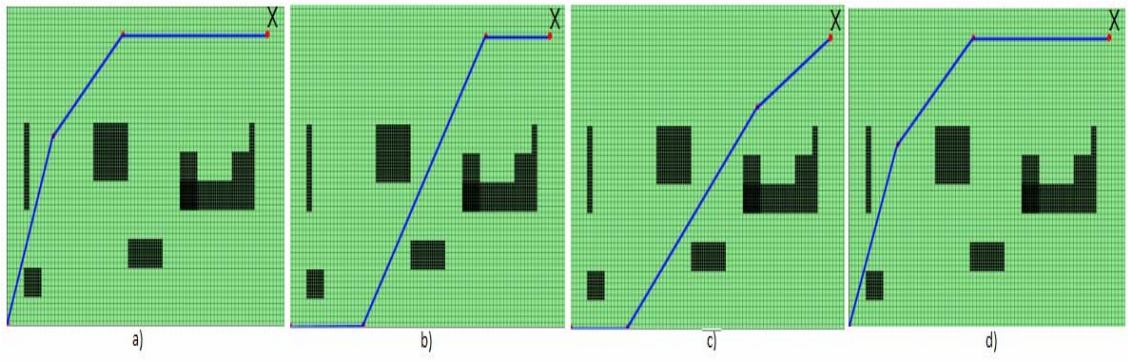

Figure 7. PSO-based solutions in a complex environment.

The information about the two environments, the initial point and the target point are given in Table 1.

Table 1: Information about the environments used in Figures 4, 5, 6 and 7.

Information	Figures 4 and 5	Figures 6 and 7
Environment	100 x 100 unit area	150 x 100 unit area
Objects	2	5
Initial point	[0, 0]	[0, 0]
Target point	[100, 100]	[150, 100]

The execution time and the length of the generated trajectories for both algorithms in all above situations are given in Table 2.The tests were performed on a computer with Intel I7 @ 2.2G Hz.

Table 2: Execution times and trajectory lengths for the two environments.

	GSA			PSO		
Figures 4 and 5		Execution time (ms)	Length $D_{\hat{i}}$		Execution time	Length $D_{\hat{i}}$
	a)	400	150.26	a)	210	155.06
	b)	410	148.28	b)	220	153.41
	c)	405	154.63	c)	225	152.84
	d)	415	148.86	d)	215	152
Figures 6 and 7	a)	680	204	a)	315	206
	b)	693	203	b)	300	201
	c)	690	198	c)	310	204
	d)	670	183	d)	320	188

The results presented in Table 2 show that the PSO-based algorithm is faster than the GSA-based one. The length paths do not differ significantly, and slightly better results are given by the GSA algorithm.

It can be noticed that the execution time is smaller for both algorithms in the simple environment shown in Figures 4 and 5. Less iterations of the algorithm are needed to be run in order for all agents to reach the target point than in the complex environment with a bigger search space from Figures 6 and 7.

The parameters of the GSA and of the PSO algorithm employed in the generation of the trajectories are given in Table 3. The notations for the parameters of the PSO algorithm are taken from (Precup et al., 2013).

Table 3: GSA and PSO parameters.

Parameters	GSA	PSO
Number of agents	100	100
Penalty (α)	1000	1000
c_1	-	1.49
c_2	-	1.49
$w_{\min}$	-	0.0001
$w_{\max}$	-	0.5
Initial velocity	[-0.2,0.2]	[-0.2,0.2]
Initial gravitational constant (G_0)	50	-
ε	0.0000000001	-

Increasing the number of agents can lead to better trajectories with the cost of the execution time. The computational complexity of both algorithms depends on the number of agents created.

An analysis of the computational complexity based on the number of agents for the simple environment (illustrated in Figure 4) is performed by its practical computation in terms of showing the execution time versus the number of agents that the algorithm is using. The results concerning the computational complexity are presented in Figure 8.

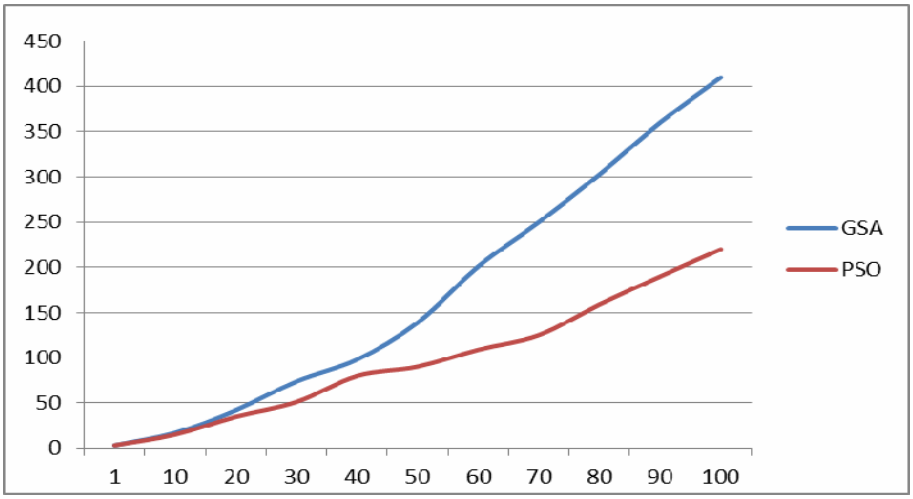

Figure 8. Execution time (ms) versus number of agents.

Several generated trajectories by the proposed GSA-based trajectory in different environments with multiple objects of different shapes are exemplified and given in Figure 9.

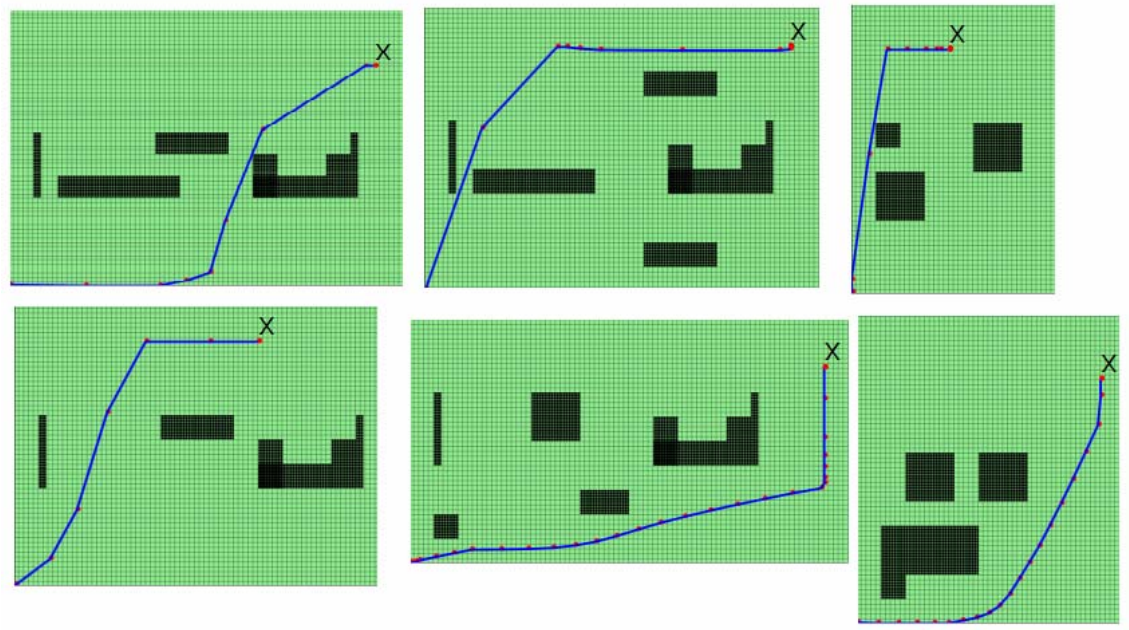

Figure 9. Several GSA-based generated trajectories.

3.2 On-line path planning algorithm

The on-line algorithm described in Subsection 2.3 is used when the robots that participate at a mission in an unknown or partially-known environment. It is assumed this way that the robots will not have collisions with any obstacles while moving to the target point.

The first example in which the online algorithm was tested is presented in Figure 10. The robot is placed at the initial point $X_0(0,0)$ and it needs to move to the target point represented by the point $X_f(100,100)$. The environment in which the robot is moving is completely unknown.

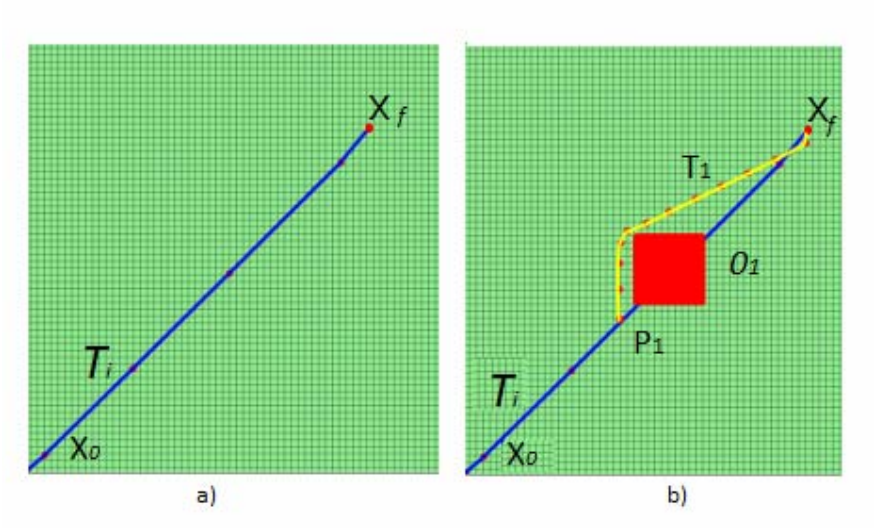

Figure 10. On-line path planning in a completely unknown environment.

Not knowing any information about obstacles the GSA-based off-line algorithm generates a trajectory from X_0 to X_f assuming that that the environment is obstacle-free. The initial off-line generated path T_i is represented by the blue trajectory in Figure 10 a). After the path is generated the robot starts to move, following the trajectory T_i. With the help of the mounted sensors the robot discovers that an object O_1 that starts at the position (50, 50) blocks the path. The robot stops its movement at the point P_1(45, 45) and updates the map with the new information. Once the map is updated, the GSA-based off-line calculates a new trajectory, this time from the point P_1 to X_f. The resulted trajectory T_1 is represented by the yellow path in Figure 10 b). Therefore the path the robot followed by the in order to travel from the initial point X_0 to the target point X_f consists of two trajectories:

- the blue trajectory that connects the X_0 to P_1,
- the yellow trajectory that connects P_1 to X_f.

Another scenario in which the on-line path planning algorithm was tested is a partially unknown environment in which the position of three obstacles (O_1, O_2, O_3) is known in advance and it is presented in Figure 11. The initial off-line trajectory is represented in the Figure 11 a) by the blue path T_i.

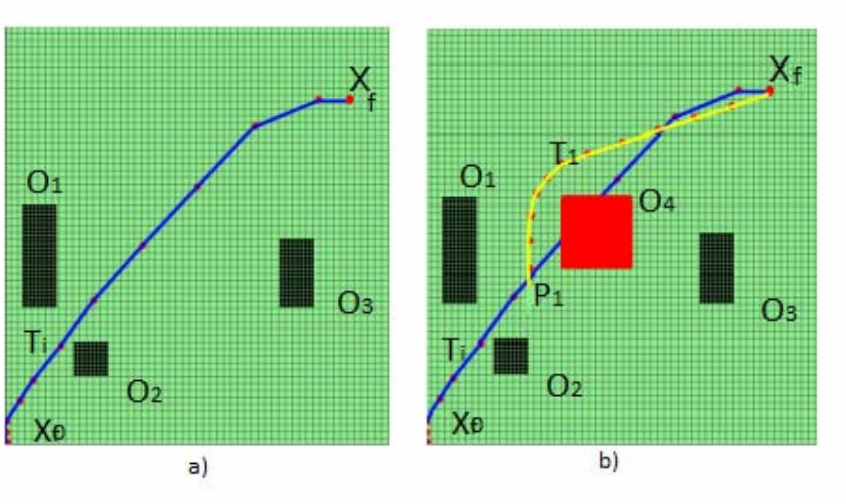

Figure 11. On-line path planning in a partially-known environment.

The recalculated path that is generated after the robot discovers that the object O_4 blocks the initial trajectory T_i is represented in the Figure 11 b) by the yellow path T_1, the final trajectory followed by the robot from X_0 to X_f is the blue path T_i from X_0 to P_1 and the yellow path T_1 from P_1 to X_f.

4. CONCLUSIONS

This paper has proposed a new algorithm for the optimal path planning. This algorithm is supported by a GSA. The proposed off-line GSA-based algorithm is able to determine optimal paths in partially-known environments where the locations of some obstacles are known in advance.

The GSA-based algorithm was compared with another evolutionary algorithm used in robot path planning represented by the PSO algorithm. It was showed that the proposed algorithm generates optimal trajectories, with path lengths that do not differ significantly by the ones generated with the PSO algorithm. Our algorithm has a slight advantage, though this comes at the cost of algorithm's execution time as the PSO algorithm is much faster.

In real world scenarios the information about the environment in which the robots are moving is unknown partially or completely, therefore an on-line path planning algorithm was designed and implemented. The on-line path algorithm starts with the generation of an off-line optimal path using only the information available in advance about the environment; this path is next used by the robot to move to the target point. Along the way to the target the robot continuously uses the sensors to determine if an obstacle blocks the initial off-line path. If there is an obstacle that blocks the path, the robot updates the map and a new off-line path is generated, this time from the new position of the robot and using the newly discovered information about the environment.

The proposed approach for the on-line path planning has the advantage of being easily generalized. It is able to employ any evolutionary algorithm for the optimal path planning generating the off-line path,

not just the GSA-based one proposed in this paper. The future research will be focused on using other evolutionary algorithms aiming the performance improvement.

ACKNOWLEDGEMENTS

This work was supported by a grant in the framework of the Partnerships in priority areas - PN II program of the Romanian National Authority for Scientific Research ANCS, CNDI - UEFISCDI, project number PN-II-PT-PCCA-2011-3.2-0732.

REFERENCES

Angelov, P.P., Lughofer, E., Zhou, Z., 2008, Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets and Systems* **159**, 23, 3160–3182.

Boonporm, P., 2011, Online geometric path planning algorithm of autonomous mobile robot in partially-known environment. *Romanian Review Precision Mechanics, Optics & Mechatronics* **40**, 185–188.

Borenstein, J., Koren, Y., 1991, The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation* **7**, 3, 278–288.

Bouhmala, N., 2012, A multilevel memetic algorithm for the satisfiability problem. *International Journal of Artificial Intelligence* **8**, S12, 78–96.

Brand, M., Masuda, M., Wehner, N., Yu, X.-H. ,2010. Ant colony optimization algorithm for robot path planning. *Proceedings of 2010 International Conference on Computer Design and Applications (ICCDA 2010)*, Qinhuangdao, Hebei, China, **3**, V3-436–V3-440.

Chen, X., Li, Y., 2006, Smooth path planning of a mobile robot using stochastic particle swarm optimization. *Proceedings of 2006 IEEE International Conference on Mechatronics and Automation*, Luoyang, Henan, China, 1722–1727

Chia, S.-H., Su, K.-L., Guo, J.-H., Chung, C.-Y., 2010, Ant colony system based mobile robot path planning. *Proceedings of 2010 4th International Conference on Genetic and Evolutionary Computing (ICGEC 2010)*, Shenzhen, China, 210–213.

Chiang, T.-A., Roy, R., 2012, An intelligent benchmark-based design for environment system for derivative electronic product development. *Computers in Industry* **63**, 9, 913–929.

Fan, X., Luo, X., Yi, S., Yang, S., Zhang, H., 2003, Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. *Proceedings of 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, Changsha, Hunan, China, **1**, 131–136.

Farahani, S.M., Abshouri, A.A., Nasiri, B., Meybodi, M.R., 2012, Some hybrid models to improve firefly algorithm performance. *International Journal of Artificial Intelligence* **8**, S12, 97–117.

Garcia, M.A., Montiel, O., Castillo, O., Sepúlveda, R., Melin, P., 2009, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. *Applied Soft Computing* **9**, 3, 1102–1110.

Gauci, M., Todd, T.j., Groß, R., 2012, Why 'GSA: a gravitational search algorithm' is not genuinely based on the law of gravity, *Natural Computing* **11**, 4, 719–720.

Gemeinder, M., Gerke, M., 2003, GA-based path planning for mobile robot systems employing an active search algorithm. *Applied Soft Computing* **3**, 2, 149–158.

Haber, R.E., del Toro, R.M., Gajate, A., 2010, Optimal fuzzy control system using the cross-entropy method. A case study of a drilling process. *Information Sciences* **180**, 14, 2777–2792.

Hermann, G., Kozlowsky, K.R., Tar, J.K., 2009, Design of a planar high precision motion stage. In: *Robot Motion and Control 2009*, K.R. Kozlowsky (Ed.), Lecture Notes in Control and Information Sciences, Springer-Verlag, Berlin, Heidelberg, **396**, 371–379.

Hu, Y., Yang, S.X., 2004, A knowledge based genetic algorithm for path planning of a mobile robot. *Proceedings of 2004 IEEE International Conference on Robotics and Automation (ICRA'04)*, New Orleans, LA, USA, **5**, 4350–4355.

Johanyák, Z.C., 2010, Survey on five fuzzy inference-based student evaluation methods. In: *Computational Intelligence in Engineering*, I.J. Rudas et al. (Eds.), Studies in Computational Intelligence, Springer-Verlag, Berlin, Heidelberg, **313**, 219–228.

Kallem, V., Komoroski, A.T., Kumar, V., 2012, Sequential composition for navigating a nonholonomic cart in the presence of obstacles. *IEEE Transactions on Robotics* **27**, 6, 1152–1159.

Kayacan, E., Cigdem, O., Kaynak, O., 2012, Sliding mode control approach for online learning as applied to type-2 fuzzy neural networks and its experimental evaluation. *IEEE Transactions on Industrial Electronics* **59**, 9, 3510–3520.

Khatib, O., 1986, Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research* **5**, 1, 90–98.

Klančar, G., Matko, D., Blažič, S., 2011, A control strategy for platoons of differential-drive wheeled mobile robot. *Robotics and Autonomous Systems* **59**, 2, 57–64.

Kovács, L., Benyó, B., Bokor, J., Benyó, Z., 2011, Induced L_2-norm minimization of glucose-insulin system for type I diabetic patients. *Computer Methods and Programs in Biomedicine* **102**, 2, 105–118.

Linda, O., Manic, M., 2011, Interval type-2 fuzzy voter design for fault tolerant systems. *Information Sciences* **181**, 14, 2933–2950.

Lozano-Perez, T., Wesley, M.A., 1979, An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM* **22**, 10, 560–570.

Lughofer, E., Macián, V., Guardiola, C., Klement, E.-P., 2011, Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems. *Applied Soft Computing* **11**, 2, 2487–2500.

Masehian, E., Sedighizadeh, D., 2010, Multi-objective PSO-and NPSO-based algorithms for robot path planning. *Advances in Electrical and Computer Engineering* **10**, 4, 69–76.

Mei, H., Tian, Y., Zu, L., 2006, A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. *International Journal of Information Technology* **12**, 3, 78–88.

Milojković, M., Nikolić, S., Danković, B., Antić, D., Jovanović, Z., 2010, Modelling of dynamical systems based on almost orthogonal polynomials. *Mathematical and Computer Modelling of Dynamical Systems* **16**, 2, 133–144.

Park, H., Kim, J.-H., 2008, Potential and dynamics-based particle swarm optimization. *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2008)*, Hong Kong, China, 2354–2359.

Pozna, C., Troester, F., Precup, R.-E., Tar, J.K., Preitl, St., 2009, On the design of an obstacle avoiding trajectory: Method and simulation. *Mathematics and Computers in Simulation* **79**, 7, 2211–2226.

Precup, R.-E., David, R.-C., Petriu, E.M., Rădac, M.-B., Preitl, S., Fodor, J., 2013, Evolutionary optimization-based tuning of low-cost fuzzy controllers for servo systems. *Knowledge-Based Systems* **38**, 74–84.

Precup, R.-E., Preitl, S., Balas, M., Balas, V., 2004, Fuzzy controllers for tire slip control in anti-lock braking systems. *Proceedings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004)*, Budapest, Hungary, **3**, 1317–1322.

Preitl, S., Precup, R.-E., 1997, *Introducere in conducerea fuzzy a proceselor*, Editura Tehnica, Bucharest.

Purcaru, C., Iercan, D., Precup, R.-E., Enache S., Dohangie, B., Fedorovici L.-O., 2012, nRobotic applications to path planning for mobile robots in missions. *Proceedings of 16th International Conference on System Theory, Control and Computing (ICSTCC 2012)*, Sinaia, Romania, 6 pp.

Qin, Y.-Q., Sun, D.-B., Li, N., Cen, Y.-G., 2004, Path planning for mobile robot using the particle swarm optimization with mutation operator. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (ICMLC 2004)*, Shanghai, China, **4**, 2473–2478.

Raja, P., Pugazhenthi, S., 2012, Optimal path planning of mobile robots: A review. *International Journal of Physical Sciences* **7**, 9, 1314–1320.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009, GSA: a gravitational search algorithm. *Information Sciences* **179**, 13, 2232–2248.

Saska, M., Macas, M., Preucil, L., Lhotska, L, 2006, Robot path planning using particle swarm optimization of Ferguson splines. *Proceedings of IEEE Conference on Emerging Technologies and Factory Automation (ETFA'06)*, Prague, Czech Republic, 833–839.

Selekwa, M.F., Dunlap, D.D., Shi, D., Collins, E.G. Jr., 2008, Robot navigation in very cluttered environments by preference-based fuzzy behaviors. *Robotics and Autonomous Systems* **56**, 3, 231–246.

Tan, G.-Z., He, H., Sloman, A., 2007, Ant colony system algorithm for real-time globally optimal path planning of mobile robots. *Acta Automatica Sinica* **33**, 3, 279–285.

Tu, J., Yang, S.X., 2003, Genetic algorithm based path planning for a mobile robot. *Proceedings of 2003 IEEE International Conference on Robotics and Automation (ICRA'03)*, Taipei, Taiwan, **1**, 1221–1226.

Tuncer, A., Yildirim, M., 2012, Dynamic path planning of mobile robots with improved genetic algorithm. *Computers & Electrical Engineering* **38**, 6, 1564–1572.

Vaščák, J., Paľa, M., 2012, Adaptation of fuzzy cognitive maps for navigation purposes by migration algorithms. *International Journal of Artificial Intelligence* **8**, S12, 20–37.

Wang, L., Liu, Y., Deng, H., Xu, Y., 2006, Obstacle-avoidance path planning for soccer robots using particle swarm optimization. *Proceedings of IEEE International Conference on Robotics and Biomimetics (ROBIO'06)*, Kunming, China, 1233–1238.

Wilamowski, B.M., Yu, H., 2010, Neural network learning without backpropagation. *IEEE Transactions on Neural Networks* **21**, 11, 1793–1803.