Journal of Intelligent & Fuzzy Systems 33 (2017) 2305–2316
DOI:10.3233/JIFS-17348
IOS Press

Optimizing robot path in dynamic environments using Genetic Algorithm and Bezier Curve

Mohamed Elhoseny[a,b,*], Abdulaziz Shehab[b] and Xiaohui Yuan[a]
[a]*Department of Computer Science and Engineering, University of North Texas, USA*
[b]*Faculty of Computers and Information Sciences, Mansoura University, Egypt*

Abstract. Robots have recently gained a great attention due to their potential to work in dynamic and complex environments with obstacles, which make searching for an optimum path on-the-fly an open challenge. To address this problem, this paper proposes a Genetic Algorithm (GA) based path planning method to work in a dynamic environment called GADPP. The proposed method uses Bezier Curve to refine the final path according to the control points identified by our GADPP. To update the path during its movement, the robot receives a signal from a Base Station (BS) based on the alerts that are periodically triggered by sensors. Compared to the state-of-the-art methods, GADPP improves the performance of robot based applications in terms of the path length, the smoothness of the path, and the required time to get the optimum path. The improvement ratio regarding the path length is between 6% and 48%. While the path smoothness is improved in the range of 8% and 52%. In addition, GADPP reduces the required time to get the optimum path by 6% up to 47%.

Keywords: Robot path planning, Bezier Curve, Genetic Algorithm, Wireless Sensor Network

1. Introduction

The advances in commercial fabrication has led robots into our daily life. The success of modern robots in applications such as Google Self-Driving Car [1] and iRobot Vacuum Cleaning robot [2] greatly depends on their ability to get the optimum routing path. The path planning problem can be described as the task a mobile robot navigation in a predefined space from a starting point A to an ending point B in such a way that avoids any obstacles [3]. Despite the great efforts to optimize the robot's path, many applications require the robot to work in dynamic and complex environments with a set of obstacles, which makes searching for an optimum robot's path on-the-fly an open challenge.

*Corresponding author. Mohamed Elhoseny, E-mails: mohamed.elhoseny@unt.edu, mohamed_elhoseny@mans.edu.eg.

Path planning has been extensively studied in industrial, civil, medical, and military environments [3–8]. In dynamic environments, obstacles have the ability to move and, hence, the map must be periodically renewed in real time accordingly. Real-time path planning algorithms should react to the changes in the environment as well as to dynamically search for an optimum path to the target. A main objective of these methods is how to find the shortest distance efficiently. In multiple-query tasks, the path planning problem gets even more complex [9]. Moreover, difficulty in path planning arises when the available data about the environment are limited [8, 10].

Many methods have been proposed to solve such a problem [4, 7, 12, 32–37], most of which are based on Heuristic search [11, 12] and pre-computation algorithms [19, 20]. Heuristic search relies on a pre-defined function that guides the search process.

Heuristic search has a low-memory overhead (linear on graph size) and adapts well to a changing graph. Pre-computation algorithms compute paths between points offline; at runtime, generating paths becomes a simple greedy search. If there is enough memory, paths can be pre-computed and all shortest paths between all nodes are stored [11, 12]. Approaches use leverage principles such as cellular automata, Dijkstra s algorithm, to plan paths [19, 20]. Others have used continuous models based on linear matrix inequalities, potential theory, and B-spline methods to create collision free paths [19, 21, 23].

The contribution of this paper is an efficient GA-based method [35, 36] for optimizing the path planning problem in dynamic environments. The proposed algorithm aims to construct an optimum collision-free trajectory from an initial location to a target location. The proposed method is called (GADPP) which uses the Bezier Curve to draw the robot path. By finding the control points which are required to draw the Bezier Curve, GADPP determines the optimum robot's path. To update the path during its movement, robot receives a signal from a BS based on alerts that are periodically triggered by sensors. GADPP utilizes 3-point interpolation by Bezier curves to generate smooth paths in the initial population in order to generate the shortest distance path in the shortest search time. GADPP works efficiently in static environments and dynamic environment where the obstacles may be mobile, appear, disappear, re-appear in real time. In addition, a run-time mechanism is developed to update the path according to the new obstacles and the shortest path found during navigation is redrawn using Bezier curve.

This paper includes four other sections. Section 2 reviews the recent works related to different path planning algorithms. Then, Section 3 explains the proposed GADPP method for path planning optimization in a dynamic environment. After that, Section 4 validate the performance of GADPP through different experiments and provides an explanation of its results. Finally, conclusion and future work are discussed in Section 5.

2. Related works

The previous works in path planning were focused on improving certain aspects of metrics using various paradigms of algorithms like heuristic algorithms or computational ones [24–28, 31]. However, these works lacked a collective evaluation and comparison of these metrics and their interdependence, which forms the base of an efficient path planning techniques. Design defects and inefficient path planning strategy in the earlier works had led to the development of newer path planning algorithms. For example, a new method based on Parallel Evolutionary Artificial Potential Field (PEAPF) in mobile robot navigation is proposed [32]. This method mainly makes it possible to deal with dynamic obstacles based on a flexible method using PEAPF. However, the required time to get the optimum path represents the main challenge to that method. An improved Distance-Bug Algorithm that comprises of two layers named deliberative layer and the reactive layer is proposed at [33]. At the first layer, the A* search is utilized to generate the desired path. The second layer directs the robot on the path generated by the first layer using distance-histogram bug algorithm that guarantee obstacle avoidance. Some drawbacks faced this method, such as it is unable to avoid obstacles with U and H shape. Moreover, it cannot be used in complex environments at which the obstacles have a high mobility feature.

Furthermore, a modified visibility graph roadmap approach that is followed by finite horizon optimal control for motion planning in an obstacle rich environment is proposed [34]. Despite the great performance to avoid the obstacles during their movement, the path smoothness represents the main problem of this method. On the other hand, a robot path planning in 3D space using a binary integer in a way that could represent the path length is proposed at [37]. Then, using binary integer programming, the path-planning problem could be solved. However, the complexity and the time consumption make this method inapplicable in many environments.

Based on its great performance in different applications, intelligent algorithms are applied to solve the path planning optimization problem as well. For example, an Intelligent Follow the Gap Method (IFGM) technique through finding the gap between obstacles is proposed [38]. Unlike other techniques, it is specifically developed to avoid obstacles with *U* and *H* shape. In addition, it could avoid local minima problem and no prior information about the environment is required. Their work was an extension to a previous work in [39]. However, IFGM needs more improvements regarding time efficiency and path smoothness. Another example is DSFCC [41] which provides a layered, dual-swarm framework with three communication channels to get the optimum path.

This method provides an efficient interaction channel for cooperation between both Wireless Sensor Network (WSN) [29, 30] and mobile multi-robot swarm. However, the processing time is the main challenge of this swarm-based intelligent method. Consequently, a WSN-aided mobile robot navigation approach is proposed [42]. This approach provides initial localization of mobile robots, orientation adjustment, path planning, and position correction with the assist of RSSI in Grid-pattern WSN. The path smoothness and the required time to get the optimum path are two big problems with that approach.

In addition, a navigation algorithm named DRAPP [40] (Distance and Robustness Aware Path Planning) which uses the RSSI- distance characteristics and audiometry are proposed to make the robot travel in the shortest geometrical path to reach the target node. However, its performance is inconsistent if it is used in the same environment. Moreover, it takes a long time to change the robot's direction in dynamic environments.

Contrary to our proposed GADPP, most of other methods proposed in the literature suffer a number of challenges, i.e., high computational times to get the optimum path and dynamic environments.

3. Methodology

Optimize the robot path in dynamic environments is not a trivial task. In a dynamic environment, a robot cannot pre-determine its path before it starts to move. For that, we assume that the working field is monitored by a WSN. WSN distributes a set of sensor nodes to track a set of targets, which are the obstacles in our case as shown in Fig. 1. These sensors are organized in network clusters and aim to collect data related to obstacles movement in the field. As soon as it receives a notification from a sensor node, the BS starts its work to update the robot path and inform it with the new one. For that, GADPP will be run at the BS time to time. In GADPP, the path of the robot is dynamically decided based on the obstacles' locations. With the goal of optimizing the distance between the start point s and the target point t, GA is employed to search for the most suitable points as the control points $\tilde{P}$ of the Bezier Curve. Using the chosen control points, the optimum path β that minimize the total distance $\tilde{D}$ between the start and the target points is selected. In the search for the suitable Bezier curve's control points, each point in the field is represented with a gene that takes 0 or 1. If one,

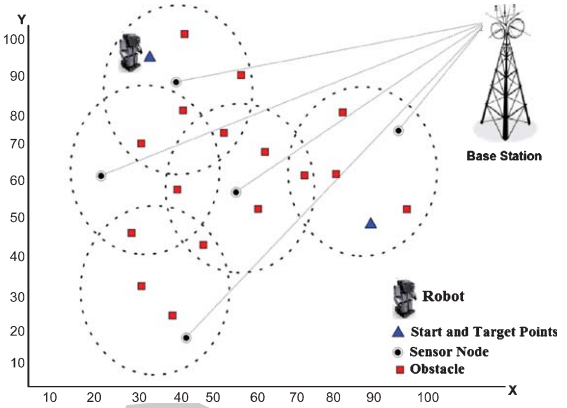

Fig. 1. An example of the working field with 16 obstacles monitored by WSN clusters.

that means that point is a control point for the Bezier curve; otherwise, it's not used in Bezier curve fitting. In addition, the cells occupied by the obstacles δ are set to be -1 and cannot be used as control points. The regular point, i.e., a point that is not an obstacle, is indicated as p. In addition, sensor nodes are responsible for monitoring the obstacles positions and inform the BS of the new position of obstacles in case of them moved.

3.1. Bezier Curve

Bézier Curves are used in computer-aided design and are named after a mathematician who used them in work for the automotive industry. One of the biggest advantages of the Bezier curve is that the Bezier curve is also convex if the control point is a convex polygon, that is, the feature polygon is convex. In addition, it can describe and express free curves and surfaces succinctly and perfectly. Thus, Bezier curve is a good tool for curve fitting, and it can be drawn as a series of line segments joining the points. These curves make "smooth" paths between two specified points. To modify the path between two points, Bézier curves specify in addition to the endpoints additional *control points*. Generally, the point x on a Bézier Curve with n control points at parameter value t can be formulated as the following:

$$\mathbf{x}(t) = \sum_{k=0}^{n} \mathbf{x}_k \binom{n}{k} t^k (1-t)^{n-k}$$

where k is the control point index. For example, let we have four control points x_0, x_1, x_2, x_3 and let $t \in \Re$. We compute a curve point with the following construction:

$$x_0^1(t) = (1-t)x_0 + tx_1$$
$$x_1^1(t) = (1-t)x_1 + tx_2$$
$$x_2^1(t) = (1-t)x_2 + tx_3$$
$$x_0^2(t) = (1-t)x_0^1(t) + tx_1^1(t)$$
$$x_1^2(t) = (1-t)x_1^1(t) + tx_2^1(t)$$
$$x_0^3(t) = (1-t)x_0^2(t) + tx_1^2(t).$$

Then, by inserting the first three equations into the next two, we obtain:

$$x_0^2(t) = (1-t)^2 x_0 + 2(1-t)tx_1 + t^2 x_2$$
$$x_1^2(t) = (1-t)^2 x_1 + 2(1-t)tx_2 + t^2 x_3.$$

Again, we insert these two equations into the last one, and after some simplifications, we obtain this result:

$$x_0^3(t) = (1-t)^3 x_0 + 3(1-t)^2 tx_1 + \Omega \quad (1)$$
$$\Omega = 3(1-t)t^2 x_2 + t^3 x_3. \quad (2)$$

x_0^3 is a point on the curve at t. Figure 2 depicts the geometric construction for $t = 1/2$.

We note that this is a cubic expression in t, so the obtained curve is a cubic curve. This curve is also affine invariant and we need 6 linear interpolations to compute a point x_0^3 on the curve. When $t = 0$, the curve is tangent to the line x_0x_1 and when $t = 1$, it is tangent to the line x_1x_2. To generate such a curve, we developed a simple algorithm that aims to subdivide the Bezier curve to a sequence of connected line segments called control polyline. A subdivision algorithm is used to recursively refine the control polyline

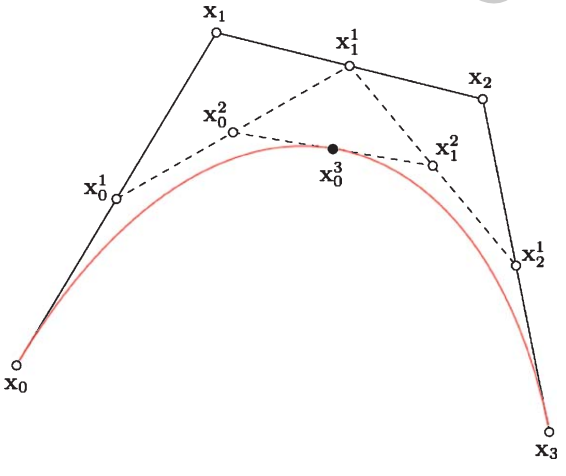

Fig. 2. Bezier Curve with four control points.

to generate progressive linear approximations to the curve. Let P_i^0 $(i = 0 \cdots n-1)$ be the original vertices of the control polyline. The following procedure generates one subdivision:

1. Define P_i^1 $(i = 1 \cdots n-1)$ as the mid-points of all line segments in the control polyline, i.e. $P_i^1 = (P_{i-1}^0 + P_i^0)/2, (i = 1 \cdots n-1)$.
2. Similarly, define P_i^2 $(i = 2 \cdots n-1)$ as the mid-points of all new line segments formed by P_i^1, i.e. $P_i^2 = (P_{i-1}^1 + P_i^1)/2, (i = 2 \cdots n-1)$.
3. Continue doing the above for each newly formed polyline, i.e. $P_i^k = (P_{i-1}^{k-1} + P_i^{k-1})/2$, where $k = 1 \cdots n-1$, and $i = k \cdots n-1$. When $k = n-1$, there is only one point left, the process is then complete.

After this subdivision, the original control polyline is divided into two separate control polylines: P_i^i $(i = 0 \cdots n-1)$, and P_{n-1}^i $(i = n-1 \cdots 0)$. Each of these two polylines represents half of the curve. But they are now closer to the curve than the original polyline.

Figure 3 shows a simulated example of the expected output of GADPP in a field of (a) 10 obstacles distributed in 100 m × 100 m area and (b) 5 obstacles distributed in 50 m × 50 m area. This figure shows how the control points affect the optimum path by forming the Bezier Curve. In Fig. 3 (a), the robot starts at (5, 43) while the endpoint is at (100, 27). There are four control points (shown as black spots in the field) at (25, 97), (43, 23), (62, 80), and (81, 5). While in Fig. 3 (b), the robot starts at the point (5, 17) while the endpoint is placed at (48, 4). There are two more control points with the start and the end points. These points are replaced at (17, 33) and (36, 48).

3.2. Problem formulation

We represent path planning as an optimization problem with a objective function with constraints as shown in Equation (5). Our objective function is to minimize the total distance $\tilde{D}$, where $\tilde{D}$ is the summation of distances d between the adjacent points $(p_1, p_2, \ldots, p_i)$ on the Bezier Curve as shown in Equation (3).

$$\tilde{D} = \sum_{i=1}^{\eta} d(p_i, p_{i+1}), \quad (3)$$

where η represents the count of the Bezier Curve's points. $\tilde{D}$ can be calculated by integral of Bezier as the follows:

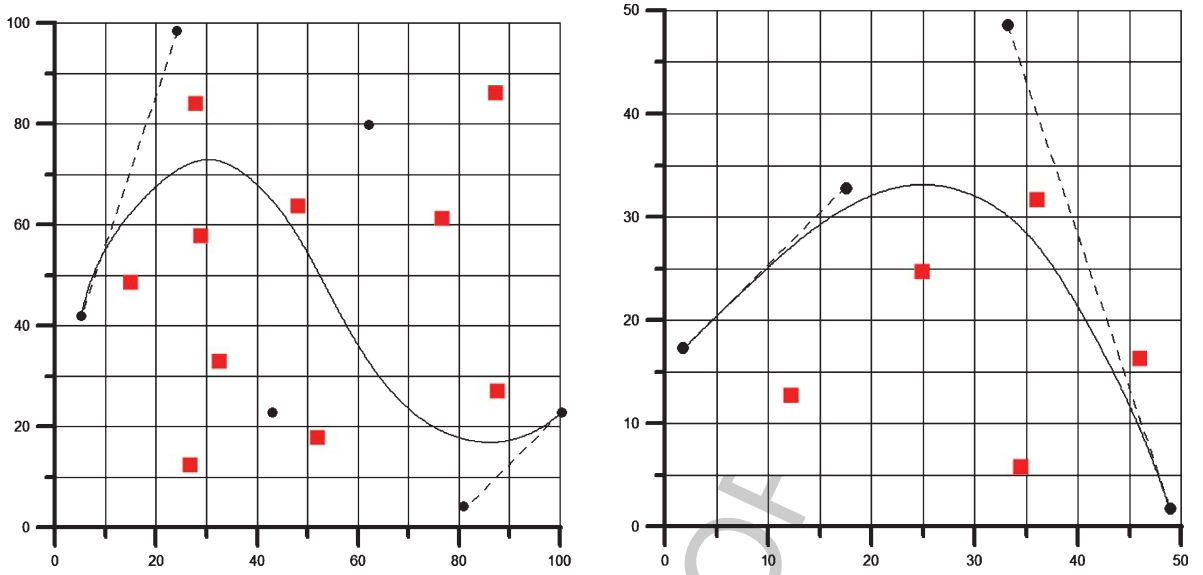

Fig. 3. An example of Bezier Curve in our working field with (a) 10 obstacles distributed in 100 m × 100 m area, and (b) 5 obstacles distributed in 50 m × 50 m area.

$$\tilde{D} = \oint_l P(t) = \oint \sum_{i=1}^{n} p_i B_{i,n}(t) = \int_0^1 \sqrt{(x_t'^2 + y_t'^2)dt}, \quad (4)$$

where l is the distance between a point p and the target t.

minimize $\tilde{D}$

subject to

$$D \geq r$$
$$\forall \theta \in \delta \Rightarrow \theta \notin \tilde{P}$$
$$\forall p \in \beta \Rightarrow p \notin \delta$$
$$s, t \notin \delta.$$
$$\tilde{P} \neq \Phi.$$

where D is the distance between two control points. δ is a set of obstacles θ.

3.3. GADPP

GA consists of three operators. **Reproduction** is the process of keeping the same chromosome without changes and transfer it to the next generation. The input and the output of this process is the same chromosome. **Crossover** is the process of concatenating two chromosomes to generate a new two chromosomes by switching genes. The input of this process is two chromosomes, while its output is two different chromosomes. A simple one-point crossover operation for binary coded populations have been used. For example, let $I = \{s_1, \ldots, s_j, \ldots, s_n\}$ and $I' = \{s'_1, \ldots, s'_j, \ldots, s'_n\}$ be two different indexes in the current population P. The crossover point was defined by randomly generating an integer j in the range $[1, n]$. Then the resulting crossed indexes are $I = \{s_1, \ldots, s_{j-1}, s'_j, \ldots, s'_n\}$ and $I' = \{s'_1, \ldots, s'_{j-1}, s_j, \ldots, s_n\}$. **Mutation** is the process of randomly reveres the value of one gene in a chromosome. So, the input is a single chromosome and the output is different one. The integer parameter to undergo a mutation, let us say s_j, is selected randomly. Then it mutates into $s'_j = 0$ if $s_j = 1$ and into $s'_j = 1$ otherwise.

GADPP consists of two fundamental components. First, a binary chromosome is used to encode the selection of control points within the entire field. Second, each chromosome is evaluated to make sure that it is valid to be a possible solution structure following the predefined constraints. The fitness is then evaluated based on this structure. The optimization goal is to minimize the distance between the start and the end points.

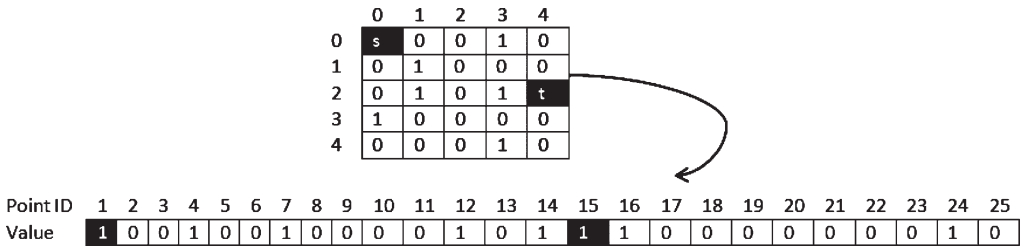

Fig. 4. Chromosome representation.

Each gene in the chromosome represents a pixel in the field. The value of a gene can be either 1 or 0, where 1 indicates that the corresponding pixel serves as the control point and 0 indicates a non-control point pixel. Figure 4 depicts a chromosome for a field with 25 nodes.

The GA's fitness function simply consists of a length of the path $\tilde{D}$. The main objective is to minimize $\tilde{D}$. The fitness function is hence defined as follows shown at Equation (5).

$$f = \frac{1}{\tilde{D}} \tag{5}$$

In addition, Equation 6 is used to calculate the probability of select (P_s) for each chromosome (ℓ). While Equation 7 calculates the expected count of select (π), and the actual count of select for each chromosome as the following:

$$P_s(i) = \frac{f(\ell)}{\sum_{i=0}^{n} f(i)} \tag{6}$$

$$\pi = \frac{f(\ell)}{[\sum_{i=1}^{n} f(i)/n]} \tag{7}$$

where n is the number of the chromosomes in the population. A new generation of the GA begins with reproduction. We select the mating pool of the next generation by spinning the weighted roulette wheel six times. So, the best chromosome representation gets more copies, the average stays even, and the worst die off and will be excluded from further processing.

Algorithm 1 summarizes the steps of GADPP. Initial values for the count of obstacles N, the count of distributed sensors on the field N_s, the start point S_P, the target point T_P, crossover ratio α, and mutation ratio β are specified. The working field is well defined by placing each sensor node s_i and obstacle o_j in different location l_i and l_j, respectively. A pool of chromosomes is randomly generated and each of them is validated to make sure that its corresponding Bezier Curve $\ddot{\beta}\kappa$ clears all obstacles. In addition, the fitness of each chromosome is calculated and the chromosome that gives the shortest path is selected.

Algorithm 1 GADPP Working Steps

1: Initialize N, N_S, S_P, T_P, α, and β
2: Generate a set of Sensors $\vec{S}$, $S = \{s_1, s_2, \ldots, s_{N_s}\}$
3: Generate a set of Obstacles $\vec{O}$, $\vec{O} = \{o_1, o_2, \ldots, o_N\}$
4: $\forall$ [$s_i \in \vec{S}$ & $o_j \in \vec{O}$] set l_i & l_j
5: Generate a pool of P chromosomes $\hat{Q} = \{q_1, q_2, \ldots, q_P\}$.
6: $\forall q_i \in \hat{Q}$, Generate a Bezier Curve $\ddot{\beta}\kappa_{q_i}$
7: $\forall\ \ddot{\beta}\kappa_{q_i}$, Validate $\ddot{\beta}\kappa_{q_i}$ using l
8: Evaluate the fitness of each $q_i \in \hat{Q}$.
9: **for** $z = 1, 2, \ldots, Z$ **do**
10: $\tilde{Q} \Leftarrow \emptyset$
11: **for** $p = 1, 2, \ldots, P$ **do**
12: Randomly select $q_a, q_b \in \hat{Q}$ $(a \neq b)$ based on the normalized fitness π
13: Cross over q_a and q_b Based on a cross point α
14: Perform mutation on q'_a and q'_b according to β
15: Evaluate $f(\tilde{q}_a)$ and $f(\tilde{q}_b)$.
16: $\tilde{\hat{Q}} \Leftarrow \tilde{\hat{Q}} \cup \{\tilde{q}_a, \tilde{q}_b\}$
17: **end for**
18: $\hat{Q} \Leftarrow \{q_i; q_i \in \tilde{\hat{Q}}$ and $f(q_i)\}$
19: **end for**
20: Return the chromosome q^* that satisfies

$$u^* = \arg\max_u f(q), q \in \hat{Q}$$

To construct the path, a Bezier Curve is drawn using the proposed control points. Then, the validation process is used to remove the invalid path, i.e., the path that intersects with an obstacle. Figure 5 illustrates the validation process that leverages the field properties. In the process of GA optimization, a new chromosome represents the proposed structure for the path. Each gene in the chromosome defines the expected role of the corresponding point, i.e., whether it serves as a control point or not. The process consults 'Obstacles', 'Distance Threshold', and 'Valid Path' sub-processes. The role of 'Obstacles' subprocess is to determine whether the point can serve as a control point (one represents serving as control point; otherwise, regular point). While the 'Distance

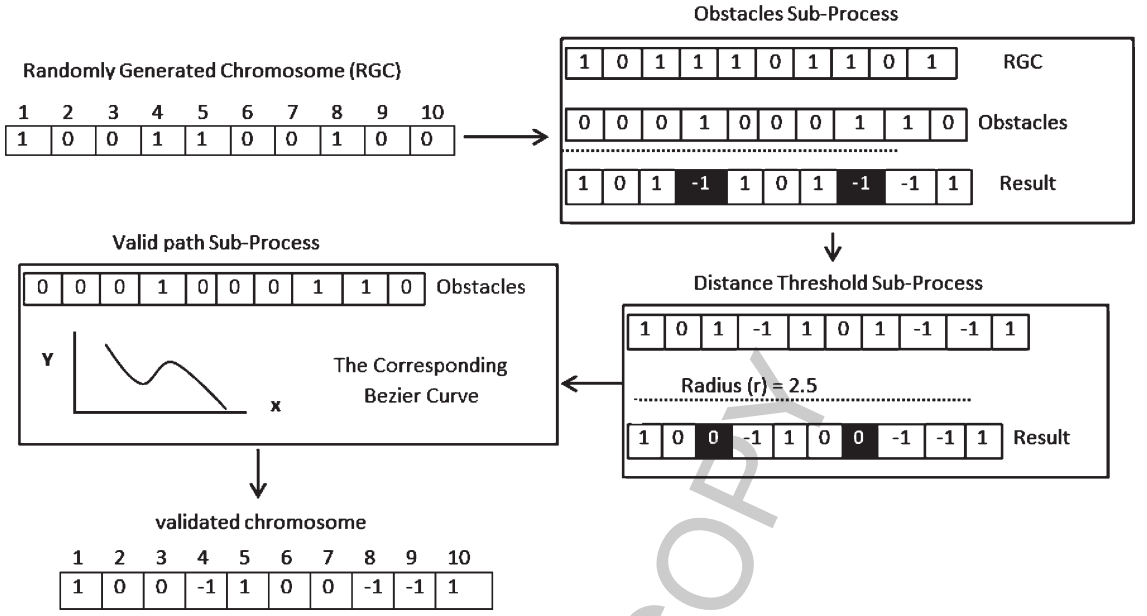

Fig. 5. An example of the validation process with 10 points, $r = 2.5$, and three obstacles.

Threshold' sub-process is used to test whether that distance between any two control points in the proposed structure is longer than the radius r or not. Then 'Valid Path' sub-process checks if the path intersects with an obstacle. The validation process determines if a chromosome complies with the constraints and hence retained in the offspring pool; otherwise, the chromosome is abandoned.

4. Experimental results and discussion

To evaluate GADPP, we used a set of well-known benchmark maps. These maps were collected from repository motion planning maps of the Intelligent and Mobile Robotics Group from the Department of Cybernetics, Czech Technical University in Prague[1].

In our experiments, ten different planar environments were used to evaluate GADPP. Figures 6 and 7 show the benchmark maps. Moreover, Table 1 lists the ID, name, and size of each map. The ID of the map is used to uniquely identify the map in our discussion. In our experiments, a dynamic environment allows to any obstacle to change its location during the robot moving. Accordingly, the Basestation runs the GA to update the robot path and notify it through the sensor nodes.

[1]Maps are available at httpimr.felk.cvut.czplanningmaps.xml

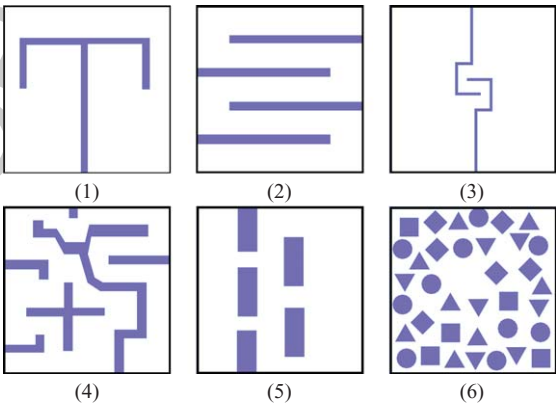

Fig. 6. Set of benchmark maps that were used in our experiment.

Tables 2, 3, and 4 list the results that compare between GADPP and the state-of-the-art methods in terms of the path length, the smoothness of the plan, and the required time to generate the optimum path using the different benchmark maps. In Table 2, the path length is the average of ten experiments for each map. The path length is calculated based on the count of points on the curve. The length is incremented by 1.5 if the robot moves diagonally. Otherwise, its value is incremented by 1. As shown, GADPP yielded the shortest length. One of the main goals of the robot path planning problems is to get the optimum smooth path. Smoothness prevents a robot from keep going

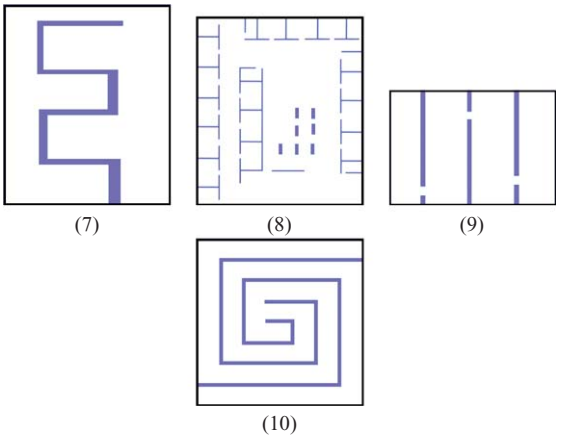

Fig. 7. Set of benchmark maps that were used in our experiment.

Table 1
Benchmark maps that are used in our experiments

ID	Name	Size ([$m \times m$])
1	T_30	10.0 × 10.0
2	back and forth	36.4 × 28.8
3	Clasp Center	20.0 × 20.0
4	Complex	20.0 × 20.0
5	Gaps	20.0 × 30.0
6	Geometry	20.0 × 20.0
7	Hidden_U	35.0 × 30.2
8	Slits_easy	36.5 × 35.0
9	square_spiral	20.0 × 20.0
10	Jari_huge	28.0 × 28.0

up and down. As a result, it reduces the required distance to arrive to the goal. In addition, it avoids the time consumption. The path smoothness can be calculated based on the number of changes in the movement directions. In our experiments, integral of squared derivative [22] is used to calculate the path smoothness. As noticed from Table 2, our proposed GADPP achieves an improvement in between 6% and 48% with respect to the second shortest path length (underlined in Table 2). Moreover, the greatest improvements was achieved in scenarios of map 9, 10, and 5. These maps have more complicated environments compared to other maps. Table 3 presents a description for path smoothness using 10 different benchmark maps. As shown in Table 3, the proposed GADPP has the highest smoothness factor compared to DRAPP, IFGM, and DSFCC. It achieves an improvement in between 8% and 52% according to the 2nd best technique (underlined at Table 3).

Table 4 lists the average time using 10 benchmark maps. Using GADPP, the time was reported at the last GA generation. The average time used by GADPP is compared with the running time of the other methods as shown in Table 4. Note that the most time-consuming process is evaluating the fitness, which can be implemented with parallel programming to improve efficiency. The improvement is in between 6% in map 8 and 45% in map 2.

A set of experiments was conducted to evaluate GADPP's performance in dynamic environments. In these experiments, different counts of obstacles are randomly placed in a field monitored by a WSN. Figure 8 shows two examples of such fields with (a) 10 obstacles distributed in 100 m × 100 m field and (b) 5 obstacles distributed in 50 m × 50 m field. In all experiments of Fig. 8(a), the start and the end points are (0,0) and (100,100), respectively and are at (0,0) and (50,50), respectively, in Fig. 8(b).

Table 5 shows the average time to get the optimum path in the dynamic environment. The second

Table 2
Path length

Map ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DRAPP	25	54	28	39	41	46	51	62	47	62
IFGM	27	50	33	29	46	42	38	71	62	59
DSFCC	22	56	41	40	37	33	42	69	52	68
GADPP	19	37	25	27	23	31	29	42	24	33
Improvement	**13%**	**26%**	**10%**	**6%**	**37%**	**6%**	**23%**	**32%**	**48%**	**44%**

Table 3
Path smoothness

Map ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
DRAPP	6	3.5	3	2.4	2.6	3.9	5	3.5	6	5.2
IFGM	7	4.6	4.8	5	3.1	3	7	4.4	7	6.7
DSFCC	6.8	5.1	3.3	2.7	2.9	4.7	5.2	4.4	4.2	5.1
GADPP	8	7.8	6.6	5.9	3.4	5.5	7.6	6.2	8.4	7.3
Improvement	**14%**	**52%**	**37%**	**18%**	**9%**	**17%**	**8%**	**40%**	**20%**	**8%**

Table 4
Run time

Map ID	1	2	3	4	5	6	7	8	9	10
DRAPP	1.50	3.62	1.92	2.09	2.54	1.91	2.74	3.01	2.80	2.94
IFGM	1.26	2.78	1.81	2.00	2.03	3.24	2.88	2.98	2.22	2.47
DSFCC	0.90	1.94	1.09	2.14	1.41	1.92	0.98	2.24	1.05	1.04
GADPP	0.52	1.05	0.75	1.05	0.94	1.27	0.79	2.10	0.79	0.76
Improvement	**40%**	**45%**	**31%**	**47%**	**33%**	**33%**	**19%**	**6%**	**24%**	**26%**

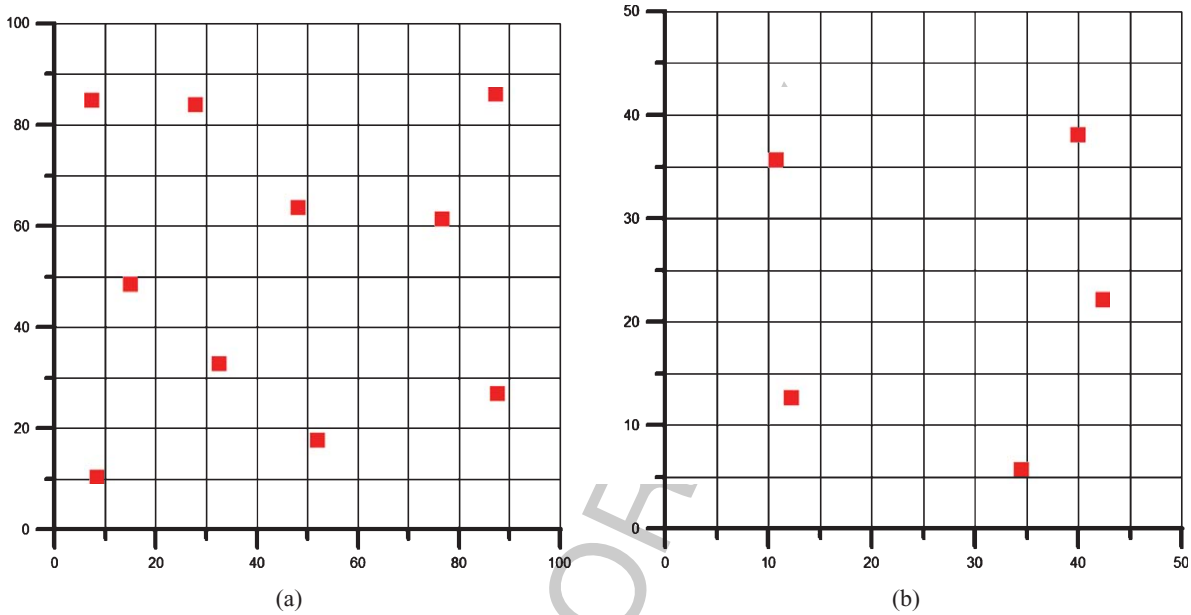

Fig. 8. Dynamic field with (a) 10 obstacle distributed in 100 m × 100 m area, and (b) 5 obstacle distributed in 50 m × 50 m area.

Table 5
Average time (Avg.) and standard deviation (STD) (in seconds) to determine the robot path using GADPP with obstacles mobility

Field	Static		Dynamic	
Area	50 m × 50 m	100 m × 100 m	50 m × 50 m	100 m × 100 m
Avg.	1.81	2.03	3.64	3.86
STD	0.22	0.19	1.66	2.01

Table 6
Path smoothness in dynamic field

Field	Static		Dynamic	
Area	50 m × 50 m	100 m × 100 m	50 m × 50 m	100 m × 100 m
Avg.	7.5	9.2	5.04	7.72
STD	0.15	0.20	0.66	0.28

raw depicts the standard deviation. Due to the obstacles mobility during the robot movement, the required time to reach the target is longer than the required time in case of a static field. Depending on the obstacles movement, the new path of the robot is determined. For that, the STD increases in dynamic environments.

Table 6 compares the performance of GADPP in a static and a dynamic environment in terms of the path smoothness. The average and the STD of ten experiments are listed. As noticed, the smoothness is decreased in the dynamic case due to the unexpected movement of the obstacles. However, the STD shows the consistency of the results.

Table 7
Path length in dynamic field

Field	Static		Dynamic	
Area	50 m × 50 m	100 m × 100 m	50 m × 50 m	100 m × 100 m
Avg.	88.5	167	109	207
STD	0.34	0.40	1.03	1.25

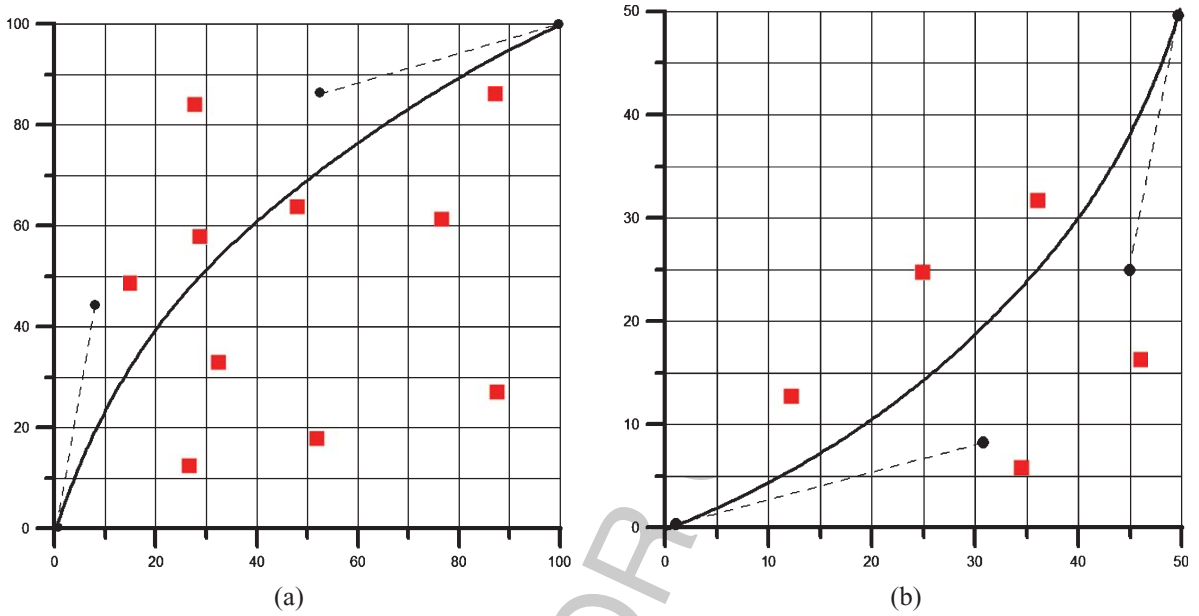

Fig. 9. An example of the optimum path given by GADPP in a static field with (a) 10 obstacles distributed in 100 m × 100 m area, and (b) 5 obstacles distributed in 50 m × 50 m area.

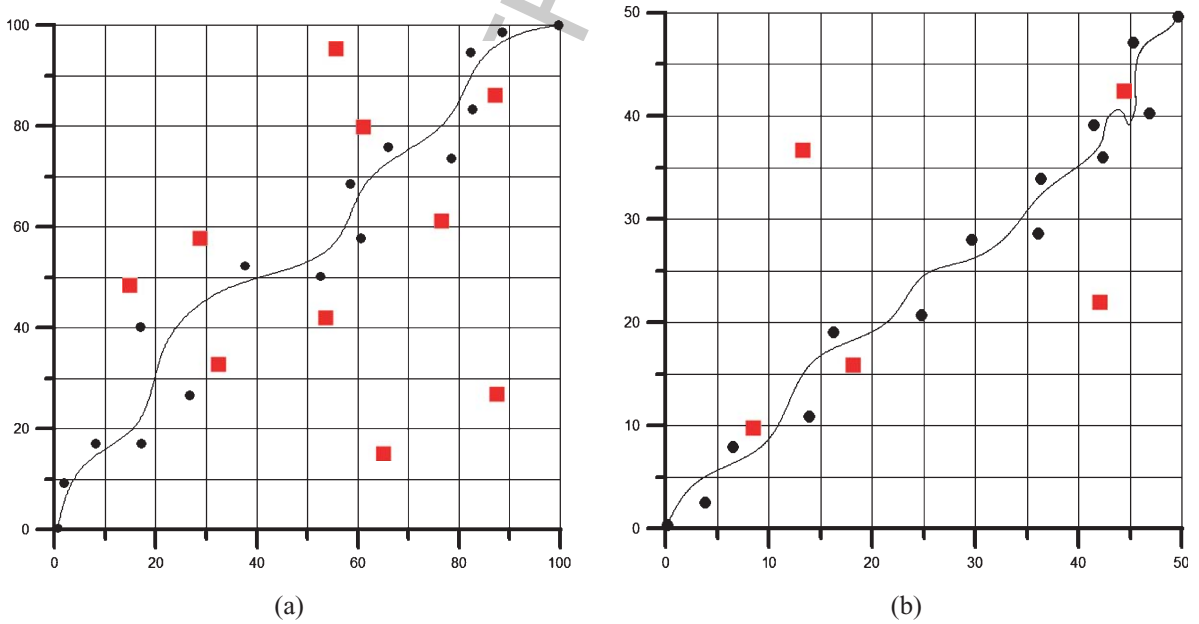

Fig. 10. An example of the optimum path given by GADPP in a dynamic field with (a) 10 obstacles distributed in 100 m × 100 m area, and (b) 5 obstacles distributed in 50 m × 50 m area.

Table 7 shows the path length for two different cases, i.e., static and dynamic environments. The path length is increased by 1.5 in the case of diagonal moving, while it is increased by one otherwise.

Figure 9 illustrates an example of the optimum solution in a static field for both of (a) field with 10 obstacles distributed in 100 m × 100 m area and (b) 5 obstacles distributed in 50 m × 50 m area. As shown in Figs. 9 (a) and 9 (b), GADPP gets the control points that make the Bezier Curve closer to the straight line to get the shortest path from the start to the target point. In addition, the paths demonstrate that GADPP generates a smooth path.

Figure 10 shows an example of a dynamic field (a) with 10 obstacles distributed in 100 m × 100 m area and (b) with 5 obstacles distributed in 50 m × 50 m area. Compared to the static fields in Fig. 9, the optimum path is a little bit longer. That is due to the change in the obstacles places during the robot movement. When obstacle moves, the path is updated accordingly based on Bezier Curve.

5. Conclusions and future work

Due to its urgent need in different applications, robots gained more attention as a research challenge. Nowadays, path planning in dynamic environments is a crucial problem faced in many robotic tasks. Real-time path planning algorithms should react to the changes in the environment as well as to dynamically search for an optimal path to the target. In this work, we proposed a new algorithm called GADPP which works in a dynamic environment to get the robot path using the Bezier Curve. The working field is monitored by a WSN to reflect any change in the environment, such as obstacle movement. The improvement ratio of GADPP regards to the path length was between 6% and 48%. While the path smoothness was improved in the range of 8% and 52%. In addition, GADPP reduced the required time to get the optimum path by 6% up to 47%. In the future, we are planning to apply our proposed GADPP method on the 3D environment. In addition, more experiments will be conducted to evaluate the performance in different applications.

References

[1] C. Chen, J. Chang and S. Chen, Collision avoidance path planning for the 6-dof robotic manipulator, in *Proceedings of the International Conference on Artificial Intelligence and Robotics and the International Conference on Automation, Control and Robotics Engineering*, New York, NY, USA, ACM, 2016, pp. 21–25.

[2] M. Shrestha, H. Yanagawa, E. Uno and S. Sugano, Efficient space utilization for improving navigation in congested environments, in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction Extended Abstracts*, New York, NY, USA, ACM, 2015, pp. 157–158.

[3] K. Miyamoto, H. Yoshioka, N. Watanabe and Y. Takefuji, Modeling of cooperative behavior agent based on collision avoidance decision process, in *Proceedings of the Second International Conference on Human-agent Interaction*, New York, NY, USA, ACM, 2014, pp. 257–260.

[4] M.S.S. Zeyad Abd Algfoor and H. Kolivand, A comprehensive study on pathfinding techniques for robotics and video games, *International Journal of Computer Games Technology* (2015).

[5] F. Yan, Y. Liu and J. Xiao, Path planning in complex 3d environments using a probabilistic roadmap method, *International Journal of Automation and Computing* **10**(6) (2013), 525–533.

[6] M. de Pinho, Z. Foroozandeh and A. Matos, Optimal control problems for path planing of auv using simplified models, in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 210–215.

[7] N.B.N.M.F.N.M.Z.A. Rastgoo and M. Naim, A critical evaluation of literature on robot path planning in dynamic environment, *Journal of Theoretical and Applied Information Technology* **70**(1) (2014), 177–185.

[8] B. Braginsky and H. Guterman, Obstacle avoidance approaches for autonomous underwater vehicle: Simulation and experimental results, *IEEE Journal of Oceanic Engineering* **41** (2016), 882–892.

[9] B. Braginsky and H. Guterman, Trajectory controller for autonomous surface vehicle under sea waves, in OCEANS 2015 - MTS/IEEE Washington, 2015, pp. 1–5.

[10] P. Grosch and F. Thomas, Geometric path planning without maneuvers for nonholonomic parallel orienting robots, *IEEE Robotics and Automation Letters* **1** (2016), 1066–1072.

[11] T.G. Shyba Zaheer, A path planning technique for autonomous mobile robot using free-configuration eigenspaces, *International Journal of Intelligent Systems and Applications in Robotics (IJRA)* (2015).

[12] E. Galceran and M. Carreras, A survey on coverage path planning for robotics, *Robotics and Autonomous Systems* **61**(12) (2013), 1258–1276.

[13] J. Yu and S.M. LaValle, Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics, *IEEE Transactions on Robotics* **32** (2016), 1163–1177.

[14] J. Keller, D. Thakur, M. Likhachev, J. Gallier and V. Kumar, Coordinated path planning for fixed-wing uas conducting persistent surveillance missions, *IEEE Transactions on Automation Science and Engineering* **14** (2017), 17–24.

[15] J. Li, G. Deng, C. Luo, Q. Lin, Q. Yan and Z. Ming, A hybrid path planning method in unmanned air/ground vehicle (uav/ugv) cooperative systems, *IEEE Transactions on Vehicular Technology* **65** (2016), 9585–9596.

[16] M. Liu, B. Xu and X. Peng, Cooperative path planning for multi-auv in time-varying ocean flows, *Journal of Systems Engineering and Electronics* **27** (2016), 612–618.

[17] E. Plaku, E. Plaku and P. Simari, Direct path superfacets: An intermediate representation for motion planning, *IEEE Robotics and Automation Letters* **2** (2017), 350–357.

[18] Z. Sun, J. Wu, J. Yang, Y. Huang, C. Li and D. Li, Path planning for geo-uav bistatic sar using constrained adaptive multiobjective differential evolution, *IEEE Transactions on Geoscience and Remote Sensing* **54** (2016), 6444–6457.

[19] S. Broumi, A. Bakal, M. Talea, F. Smarandache and L. Vladareanu, Applying dijkstra algorithm for solving neutrosophic shortest path problem, in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2016, pp. 412–416.

[20] P.G. Tzionas, A. Thanailakis and P.G. Tsalides, Collisionfree path planning for a diamond-shaped robot using two-dimensional cellular automata, *IEEE Transactions on Robotics and Automation* **13** (1997), 237–250.

[21] M. Jiang, Y. Chen, W. Zheng, H. Wu and L. Cheng, Mobile robot path planning based on dynamic movement primitives, in *2016 IEEE International Conference on Information and Automation (ICIA)*, 2016, pp. 980–985.

[22] S. Ovchinnikov, Measure, Integral, Derivative: A Course on Lebesgue's Theory, Book, Springer, 2013.

[23] W. Huang, A. Osothsilp and F. Pourboghrat, Vision-based path planning with obstacle avoidance for mobile robots using linear matrix inequalities, in *2010 11th International Conference on Control Automation Robotics Vision*, 2010, pp. 1446–1451.

[24] H.H. Triharminto, T.B. Adji and N.A. Setiawan, Dynamic uav path planning for moving target intercept in 3d, in *2011 2nd International Conference on Instrumentation Control and Automation*, 2011, pp. 157–161.

[25] A. Majumdar, A.A. Ahmadi and R. Tedrake, Control design along trajectories with sums of squares programming, in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 4054–4061.

[26] A. Chamseddine, Y. Zhang, C.A. Rabbath, C. Join and D. Theilliol, Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle, *IEEE Transactions on Aerospace and Electronic Systems* **48** (2012), 2832–2848.

[27] A. Shariati, A. Ghaffari and A.H. Shamekhi, Paths of twowheeled self-balancing vehicles in the horizontal plane, in *2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*, 2014, pp. 456–461.

[28] Y.B. Jia, Planning the initial motion of a free sliding/rolling ball, *IEEE Transactions on Robotics* **32** (2016), 566–582.

[29] X. Yuan, M. Elhoseny, H. Minir and A. Riad, A genetic algorithm-based, dynamic clustering method towards improved WSN longevity, *Journal of Network and Systems Management, Springer US* **25**(1) (2017), 21–46. DOI: 10.1007/s10922-016-9379-7

[30] M. Elhoseny, X. Yuan, Z. Yu, C. Mao, H.K. El-Minir and A.M. Riad, Balancing energy consumption in heterogeneous wireless sensor networks using genetic algorithm, *IEEE Communications Letters* **19**(12) (2015), 2194–2197. DOI: 10.1109/LCOMM.2014.2381226

[31] J. Yuan, F. Sun and Y. Huang, Trajectory generation and tracking control for double-steering tractor-trailer mobile robots with on-axle hitching, *IEEE Transactions on Industrial Electronics*, 7665–7677.

[32] O. Montiel, R. Seplveda and U. Orozco-Rosas, Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field, *Journal of Intelligent & Robotic Systems* **79**(2) (2015), 237–257.

[33] Y. Zhu, T. Zhang, J. Song and X. Li, A new hybrid navigation algorithm for mobile robots in environments with incomplete knowledge, *Knowledge-Based Systems* **27** (2012), 302–313.

[34] S.H. Kim and R. Bhattacharya, Multi-layer approach for motion planning in obstacle rich environments read more: http://arc.aiaa.org/doi/abs/10.2514/6.2007-6603, in AIAA Guidance, Navigation and Control Conference and Exhibit, Guidance, Navigation, and Control and Co-located Conferences, 2007.

[35] N. Metawa, M. Elhoseny, M. Kabir Hassan and A. Hassanien, Loan portfolio optimization using genetic algorithm: A case of credit constraints, *12th International Computer Engineering Conference (ICENCO), IEEE*, 2016, pp. 59–64. DOI: 10.1109/ICENCO.2016.7856446

[36] N. Metawa, M. Kabir Hassan and M. Elhoseny, Genetic algorithm based model for optimizing bank lending decisions, *Expert Systems with Applications* **80** (2017), 75–82. ISSN 0957-4174, http://dx.doi.org/10.1016/j.eswa.2017.03.021

[37] G. Habibi, E. Masehian and M.T.H. Beheshti, Binary integer programming model of point robot path planning, in *IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society*, 2007, pp. 2841–2845.

[38] M. Zohaib, S.M. Pasha, N. Javaid, A. Salaam and J. Iqbal, An improved algorithm for collision avoidance in environments having u and h shaped obstacles, *Studies in Informatics and Control* **23**(1) (2014), 23.

[39] M. Pasha, R.A. Riaz, N. Javaid, M. Ilahi and R.D. Khan, Control strategies for mobile robot with obstacle avoidance, *Journal of Basic and Applied Scientific Research* **3**(4) (2013), 1027–1036.

[40] Z. Zhang, Z. Li, D. Zhang and J. Chen, Path planning and navigation for mobile robots in a hybrid sensor network without prior location information, *International Journal of Advanced Robotic Systems* **10**(3) (2013), 172.

[41] W. Li and W. Shen, Swarm behavior control of mobile multirobots with wireless sensor networks, *Journal of Network and Computer Applications* **34**(4) (2011), 1398–1407. Advanced Topics in Cloud Computing.

[42] N. Zhou, X. Zhao and M. Tan, Rssi-based mobile robot navigation in grid-pattern wireless sensor network, in *2013 Chinese Automation Congress*, 2013, pp. 497–501.