Path Planning for an HEXA Parallel Mechanism using a Modified PSO Algorithm

Lucian Milica (milica.lucian@ugal.ro)
"Dunarea de Jos" University of Galati https://orcid.org/0000-0003-0293-9428
Adina Milica
Alexandru Ioan Cuza University: Universitatea Alexandru Ioan Cuza

Original Article

Keywords: Mobile Robot Motion Planning, Optimization Metods, Parallel Arhitectures, Mobile Robot Kinematics

Posted Date: November 15th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1038719/v1

Path planning for an HEXA parallel mechanism using a modified PSO algorithm

Lucian Milica[1] and *Adina* Milica[2]
[1]"Dunărea de Jos" University of Galați, Faculty of Engineering,
Domnească no. 47, Galați, 800201, Romania
[2]"Alexandru Ioan Cuza" University of Iași, Faculty of Mathematics,
Bulevardul Carol I no.11, Iaşi, 700506, Romania

Correspondent author email: milica.lucian@ugal.ro

Keywords: Mobile Robot Motion Planning, Optimization Metods, Parallel Arhitectures, Mobile Robot Kinematics

Abstract

This paper presents a method for determining the optimal trajectory of the characteristic point based on the kinematic analysis of a HEXA parallel mechanism. The optimization was performed based on a modified PSO algorithm based on Hermite polynomials (MH-PSO). The change made to the initial algorithm consists in restricting the search space of the solutions by using the Hermite polynomial expressions of the geometric parameters as time functions for defining the movements of the end-effector. The MH-PSO algorithm, from its inception, ensures a faster convergence of solutions and ease of computational effort and is the main advantage of the method presented. During the optimization process, the function followed was the length of the trajectory described by the sequence of positions of the characteristic point, belonging to the end effector element, in compliance with additional conditions imposed. The use of the Hermite functions and PSO algorithm leads to minimal effort for analysis and mathematical formulation of the optimization problem.

1. Introduction

Optimization in the design process is a mathematical analysis tool, used to determine the optimal solutions representing all the complex problems of this process. In general, the basic elements of an optimization problem are the objective function, the design variables, the constraints and the limits within the variables can take values. The objective function is the mathematical expression that represents the purpose of optimization. The optimization process actually aims to minimize or maximize this function. The design variables are the unknowns of the optimization problem to be determined. These are numeric values that correspond to the value of the objective function. In the optimization process, constraints are a set of conditions that must be met for this process to be feasible. Constraints allow design variables to take certain values and exclude others. The limits on which variables can take values limit the range in which they can take values and, together with the constraints, are used to determine whether solutions are feasible or impossible to achieve. When considering the robot's route planning, the main objectives pursued in the optimization process are: finding a trajectory that avoids collisions, from the starting position to the final position; the time during which the movement is performed; characteristic point speed. To solve the optimization problem, a number of researchers undertook a series of studies in which heuristic algorithms were used. Heuristic techniques are an alternative way of solving optimization problems that are difficult to solve by classical methods [1]. The usefulness of heuristic search algorithms has been demonstrated for problems for which either there is no mathematical model of direct solving, or this model is too complicated to implement. The range of applications of heuristic algorithms is quite varied, especially since most real-life problems start from these premises. Solving an optimization problem using heuristic algorithms means determining the values of the design variables to minimize the *objective function* (FO) depending on the constraints imposed. If the solution found does not satisfy the imposed constraints, it is not accepted even if the value of the FO function is minimal. The algorithm adds a *penalty function* (FP) to the FO function. The two functions (FP and FO) form the *fitness function* (FF). There are two situations in this way:

- if the constraints are in the feasible zone then FP = 0;

- if the constraints are not in the feasible area then the FF function is penalized by the FP function.

Generally, an algorithm starts with a number of randomly generated initial solutions. These solutions represent the initial values of the design variables. The value of the FO function is calculated with these initial values. If the initial values satisfy the constraints, the FF function will be the same as the FO function. The second set of solutions is generated, with values close to the initial solutions. Between the two FF functions, the one with the minimum value is designated as the preferred solution. As long as the iteration continues, the algorithm converges to the global minimum value [2]. An important problem in the case of heuristic algorithms is the local minimums which represent a blockage for them. Various strategies have been developed to avoid this blockage. In the first iterations, the algorithm looks for global solutions with large steps to avoid local minimums and in the following iterations, it looks for solutions with small steps to find the best minimum. Another way to solve optimization problems is with the help of *particle swarm optimization* (PSO). The concept of PSO was introduced by James Kennedy and Russell Eberhart in 1995 representing at that time a revolutionary computing technique inspired by the social interaction between insects or animals. Like *genetic algorithms* (GA), PSO mimics the natural biological evolution, on the principle of survival of the fittest in order to obtain the best solution. In general, heuristic algorithms do not seek to find an optimal solution but try to find an acceptable solution near the optimal solution within a reasonable time. PSO is simpler than GA because it does not have evolution operators such as "crossover" and "mutation" as in the latter [3]. As an optimization algorithm, PSO is less complex and more efficient compared to other optimization algorithms [4-7]. Among the advantages can be listed: simplicity, fast convergence, some parameters to be adjusted, and no operator [8, 9]. PSO is applied for a wide range of applications, such as function optimization, power system applications, planning problems in research operations, and for a number of robotics applications [10, 11].

More simulation results shows the applicability of PSO in path planning in static or dynamic environments [12,13]. Also, a multi-objective particle swarm optimization algorithm is applied to minimize the travel time, energy and the distance of the end-effector [14].

2. Polynomial formulation of the motion parameters for the HEXA parallel mechanism

One of the k kinematic chain scheme of the HEXA parallel mechanism is presented in Fig.1. Each of thease chain has the kinematic revolute joint at point D_k and the kinematic spherical joints at points E_k and F_k respectively. The actuating arms r_k are rotated at an angle θ_k around the axis of the unit vector $\mathbf{u_k}$ passing through the point D_k.

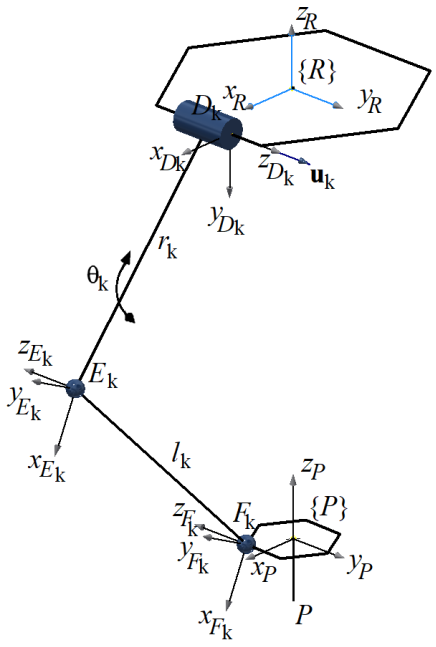

Fig 1. The kinematic scheme

The HEXA mechanism has the following particularities:

1. axis of rotational active joints by $\mathbf{u_k}$ unit vector are two-by-two coincidentand, coplanar and central-symmetrically disposed;
2. actuating arms r_k are of equal length;
3. the rods l_k are of equal length;
4. centers of spherical joints F_k are coplanar and located on the vertices of a regular hexagon;

The coordinates of the successive positions of the characteristic point P_j are time functions and have the expressions:

$$x = x(t);\ y = y(t);\ z = z(t); \tag{1}$$

Based on Hermite polynomials each of the three equations can be formed. A polynomial function having the expression (2) will respect the same conditions at the ends of a range $[q_1, q_2]$:

$$H(q) = H_1(q) + H_2(q) + H_3(q) + H_4(q) + H_5(q) + H_6(q) \tag{2}$$

The functions H_j will be defined by a linear transformation of some more general $h_j(\lambda)$ functions such that:

$$H_j(q) = c_j \cdot h_j(\lambda) \tag{3}$$

where λ is the new independent variable defined by linear transformation:

$$\lambda = \frac{q-q_1}{q_2-q_1} \tag{4}$$

The equation (2) becomes:

$$H(q) = \sum_{j=1\ldots6} c_j \cdot h_j(\lambda) \tag{5}$$

The equation of motion of the point projected on the three axes becomes:

$$x(t) = x_1 \cdot h_1(\lambda) + x_2 \cdot h_2(\lambda) + x_1' \cdot T \cdot h_3(\lambda) + x_2' \cdot T \cdot h_4(\lambda) + x_1'' \cdot T^2 \cdot h_5(\lambda) + x_2'' \cdot T^2 \cdot h_6(\lambda) \tag{6}$$

$$y(t) = y_1 \cdot h_1(\lambda) + y_2 \cdot h_2(\lambda) + y_1' \cdot T \cdot h_3(\lambda) + y_2' \cdot T \cdot h_4(\lambda) + y_1'' \cdot T^2 \cdot h_5(\lambda) + y_2'' \cdot T^2 \cdot h_6(\lambda) \tag{7}$$

$$z(t) = z_1 \cdot h_1(\lambda) + z_2 \cdot h_2(\lambda) + z_1' \cdot T \cdot h_3(\lambda) + z_2' \cdot T \cdot h_4(\lambda) + z_1'' \cdot T^2 \cdot h_5(\lambda) + z_2'' \cdot T^2 \cdot h_6(\lambda) \tag{8}$$

The coefficients $x_1 \ldots x_2''$ of the functions $h_j(\lambda)$ represent the values of the coordinates of the fixed points of the extremities of the interval respectively their derivatives dr/dt and d^2r/dt^2. On the interval AB will have two times t_A and t_B corresponding to the moment of time when P_j is identical with point A respectively with point B. We considered $t_A = 3[s]$, $t_B = 8[s]$. T represents the time of motion ($t_{end} - t_{start}$) on a certain interval.

The coefficients $x_1 \ldots x_2''$ has the expressions:

$$\begin{gathered} x' = s'cos(\varphi_1) \cdot cos(\psi_1);\ y' = s'cos(\psi_1) \cdot sin(\varphi_1);\ z' = s'sin(\psi_1); \\ x'' = s''cos(\varphi_2) \cdot cos(\psi_2);\ y'' = s''cos(\psi_2) \cdot sin(\varphi_2);\ z'' = s''sin(\psi_2). \end{gathered} \tag{9}$$

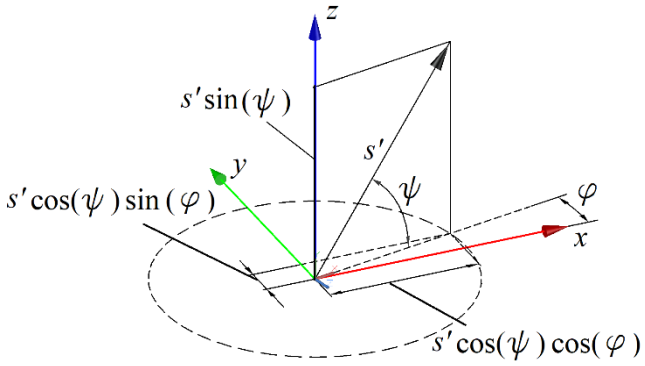

Fig. 2 Representation of angles φ and ψ

The directions of velocities and accelerations of the extreme points of the interval AB is given by the angles $\varphi_{1,2}$ and $\psi_{1,2}$ (Fig. 2).

The following notation was done:

$$\begin{aligned}
&\left|v_P^A\right| = s'_A;\ \left|v_P^B\right| = s'_B;\ \left|v_P^C\right| = s'_C;\\
&\left|a_P^A\right| = s''_A;\ \left|a_P^B\right| = s''_B;\ \left|a_P^C\right| = s''_C;\\
&x'(A) = v_{Px}^A\ ;\ y'(A) = v_{Py}^A;\ z'(A) = v_{Pz}^A;\\
&x'(B) = v_{Px}^B\ ;\ y'(B) = v_{Py}^B;\ z'(B) = v_{Pz}^B;\\
&x''(A) = a_{Px}^A;\ y''(A) = a_{Py}^A;\ z''(A) = a_{Pz}^A;\\
&x''(B) = a_{Px}^A;\ y''(B) = a_{Py}^B;\ z''(B) = a_{Pz}^B;
\end{aligned} \tag{10}$$

Deriving the equation (6,7,8) to time and taking into account that:

$$\dot{h}_j(\lambda) = \frac{1}{T}\cdot h'_j(\lambda) \text{ and } \ddot{h}_j(\lambda) = \frac{1}{T^2}\cdot h''_j(\lambda) \tag{11}$$

the expressions of speed and acceleration of the P_{j_x} point as exemple will be:

$$v_x(t) = \frac{x_1}{T}\cdot h'_1(\lambda) + \frac{x_2}{T}\cdot h'_2(\lambda) + x'_1\cdot h'_3(\lambda) + x'_2\cdot h'_4(\lambda) + x''_1\cdot T\cdot h'_5(\lambda) + x''_2\cdot T\cdot h'_6(\lambda) \tag{12}$$

$$a_x(t) = \frac{x_1}{T^2}\cdot h''_1(\lambda) + \frac{x_2}{T^2}\cdot h''_2(\lambda) + \frac{x'_1}{T}\cdot h''_3(\lambda) + \frac{x'_2}{T}\cdot h''_4(\lambda) + x''_1\cdot h''_5(\lambda) + x''_2\cdot h''_6(\lambda) \tag{13}$$

3. The optimal trajectory obtained based on modified PSO algotithm

The orientation of the platform is given by the angles $\alpha = \beta = 0°, \gamma =$ constant. The three angular parameters α, β, γ represent Euler's angles in the Roll-Pitch-Yaw system and describe the orientation of the platform.

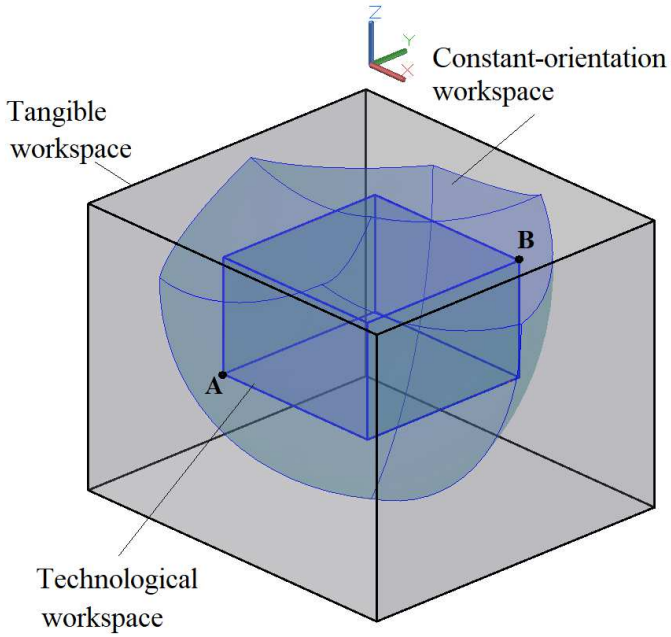

Fig. 3 The graphic representation of the three spaces

The point P_j moves between points $A(-150,-150,-494)$ and $B(150,150,-308)$ inside the technological space of the robot whose limits are defined by the conditions: $x_{P_j} \in [-150,150]$, $y_{P_j} \in [-150,150]$, $z_{P_j} \in [-494,-308]$. The volume V_{th} of the technological space ($V_{th} = 16.71\ \mathrm{dm}^3$), is a subspace of the volume of the workspace of constant orientation, included in the tangible space V_{tan} of the HEXA mechanism. This volume has the value $V_{tan} = 155.12\ \mathrm{dm}^3$ and can be contained in the volume

of a rectangular parallelepiped. The sequence of positions P_j in this space is restricted by the following conditions: $x_{P_j} \in [-266, 309.7]$, $y_{P_j} \in [-314.5, 280.8]$, $z_{P_j} \in [-520.4, -150.3]$ (Fig . 3).

Let points A and B belong to V_{th}. We propose that starting from the Hermite functions $f_i(\lambda)$, based on the PSO algorithm to optimize the length of the trajectory traveled by the characteristic point on the interval AB. The algorithm is initiated by determining the position of the points P_j on the curve on the interval AB by means of the functions $f_i(\lambda)$ whose expression was established on the basis of the relations (6-8).

$$\begin{aligned} f_x(\lambda) &= 1750.25 \cdot \lambda^5 - 4291 \cdot \lambda^4 + 2840.75 \cdot \lambda^3 - 150; \\ f_y(\lambda) &= 1144.875 \cdot \lambda^5 - 3080.25 \cdot \lambda^4 + 2235.375 \cdot \lambda^3 - 150; \\ f_z(\lambda) &= 728.55 \cdot \lambda^5 - 1793.95 \cdot \lambda^4 + 1251.4 \cdot \lambda^3 - 494; \end{aligned} \tag{14}$$

where $\lambda \in [0,1]$.

We impose certain kinematic conditions on the ends of the interval (in points A and B), as well as the start and end times of the motion $t_A = 3\ s$, $t_B = 8\ s$. Based on the functions $f_i(\lambda)$ we determine a set of points P_j and display their coordinates. These points represent *best local positions* (BLP) for the points P_j and we will denote $\mathbf{p}_j^k = \left(p_{j1}^k,\ p_{j2}^k,\ p_{j3}^k\right)$.

The algorithm continues with the solutions found in the previous step to which we impose new conditions: $\left|\mathbf{v}_{\mathbf{P_j}}\right| \in [0, 450]$. We will denote by $\mathbf{g}_j^k = \left(g_{j1}^k,\ g_{j2}^k,\ g_{j3}^k\right)$ the position vectors of the points P_j that satisfy these conditions. These points represent *best global positions* (BGP) for the points P_j. For $\mathbf{g}_j^k$ we display their velocities which we denote $\mathbf{v}_j^k = \left(v_{j1}^k,\ v_{j2}^k,\ v_{j3}^k\right)$.

Using the *Rand* function we generate the successive positions of the P_j points and exclude those points that do not meet the conditions: $x_{P_j} \in [-150, 150]$, $y_{P_j} \in [-150, 150]$, $z_{P_j} \in [-494, -308]$. We display the coordinates of the points. We will denote by $\mathbf{x}_j^k = \left(x_{j1}^k,\ x_{j2}^k,\ x_{j3}^k\right)$ the position vectors of these points. This points represents the initial condition by which we restrict the search for solutions among the set of points that belong to the technological workspace.

Next we call the objective function L_t which calculates the length of the trajectory described by the sequence of points P_j based on their coordinates. The expression of this function is:

$$L_t = \sum s_{j+1,j}\,; \tag{15}$$

$$s_{j+1,j} = \sqrt{\left(x_{P_{j+1}} - x_{P_j}\right)^2 + \left(y_{P_{j+1}} - y_{P_j}\right)^2 + \left(z_{P_{j+1}} - z_{P_j}\right)^2}\,; \tag{16}$$

The program compares the value found for L_t with a value imposed for it, denoted $L_{t_i} = 600\ mm$. The value imposed for L_{t_i} can be any other value depending on the design requirements. If the limit is reached, the program stops and displays the coordinate values of the points P_j for which the condition is met. If the condition is not met, the program proceeds to the next step. In the next step, the speeds are updated according to equation (17).

$$\mathbf{v}_j^{k+1} = \omega \cdot \mathbf{v}_j^k + c_1 \cdot \mathbf{R}_1^k \cdot \left(\mathbf{p}_j^k - \mathbf{x}_j^k\right) \cdot \alpha(t) + c_2 \cdot \mathbf{R}_2^k \cdot \left(\mathbf{g}_j^k - \mathbf{x}_j^k\right) \cdot \alpha(t) \tag{17}$$

where:

- $\mathbf{v}_j^k$, $\mathbf{v}_j^{k+1}$- represent the speed of P_j point at k iteration respectively at $k+1$ iteration;
- $\mathbf{x}_j^k$- represent the position vector of P_j point at k iteration;
- $\mathbf{p}_j^k$- represent BLP for P_j point at k iteration;
- $\mathbf{g}_j^k$- represent BGP for P_j point at k iteration;
- c_1, c_2- are two constants representing the cognitive and social parameters (or acceleration parameters) that determine the convergence of solutions to the best local and global positions. Generally the constants must satisfy the condition, $c_1 + c_2 \leq 4$ [15];

- $\mathbf{R}_1^k$, $\mathbf{R}_2^k$- are two matrices by 3×3 dimension, randomly generated in range interval [0,1] for each P_j point at each k iteration
- ω- constant of inertia [16]. The expression of this parameter has been discussed in various specialized papers and its value is in the range [0.4, 0.9];
- $\alpha(t) = \frac{1}{(t_B - t_A)}$.

Recent experiments have shown that for a series of values of the parameters of acceleration and inertia, the velocity vectors tend to infinity producing the swarm explosion. The causes of these divergences have been analyzed by a number of researchers [17-19] who have concluded that better stability of the system is given by the assignment of the following values for the acceleration coefficients and the inertia coefficient:

$$c_1 = c_2 = 1.7, \omega = 0.6 \quad (18)$$

$$c_1 = c_2 = 1.7, \omega = 0.715 \quad (19)$$

$$c_1 = c_2 = 1.55, \omega = 0.42 \quad (20)$$

In this paper we considered the values from relation (19) for the acceleration and the inertia coefficient. With these notations equation (17) becomes:

$$\mathbf{v}_j^{k+1} = 0.715 \cdot \mathbf{v}_j^k + 1.7 \cdot \mathbf{R}_1^k \cdot \left(\mathbf{p}_j^k - \mathbf{x}_j^k\right) \cdot \alpha(t) + 1.7 \cdot \mathbf{R}_2^k \cdot \left(\mathbf{g}_j^k - \mathbf{x}_j^k\right) \cdot \alpha(t) \quad (21)$$

In the next step, based on the speeds updated in the previous step, we will update the positions of the P_j points with relation (22):

$$\mathbf{x}_j^{k+1} = a \cdot \mathbf{x}_j^k + \cdot \beta(t) \cdot \mathbf{v}_j^{k+1} \quad (22)$$

- $\mathbf{x}_j^{k+1}$- represent the position vector of P_j point at $k+1$ iteration;
- $a, \beta(t)$ are convergence parameters. The values for the two parameters are set as follows: $a = 1$, $\beta(t) = t_j$, where t_j represents the time corresponding to the position P_j at the iteration $k+1$.

In this new stage the points $\mathbf{x}_j^{k+1}$ are displayed and based on them the new value of the function L_t is calculated by setting the condition $L_{t_j}^{\,k} < L_{t_j}^{\,k+1} \leq L_{t_i}$.

The algorithm continues with the successive updating of the velocities and positions of the P_j points:

$$\mathbf{v}_j^{k+2} = 0.715 \cdot \mathbf{v}_j^{k+1} + 1.7 \cdot \mathbf{R}_1^k \cdot \left(\mathbf{g}_j^k - \mathbf{p}_j^k\right) \cdot \alpha(t) + 1.7 \cdot \mathbf{R}_2^k \cdot \left(\mathbf{x}_j^{k+1} - \mathbf{p}_j^k\right) \cdot \alpha(t) \quad (23)$$

$$\mathbf{x}_j^{k+2} = a \cdot \mathbf{p}_j^k + \beta(t) \cdot \mathbf{v}_j^{k+2} \quad (24)$$

The points $\mathbf{x}_j^{k+2}$ are displayed and based on them the new value of the function L_t is calculated by setting the condition $L_{t_j}^{\,k+1} < L_{t_j}^{\,k+2} \leq L_{t_i}$. The speeds and positions of the P_j points are updated successively as above:

$$\mathbf{v}_j^{k+3} = 0.715 \cdot \mathbf{v}_j^{k+2} + 1.7 \cdot \mathbf{R}_1^k \cdot \left(\mathbf{x}_j^{k+1} - \mathbf{g}_j^k\right) \cdot \alpha(t) + 1.7 \cdot \mathbf{R}_2^k \cdot \left(\mathbf{x}_j^{k+2} - \mathbf{g}_j^k\right) \cdot \alpha(t) \quad (25)$$

$$\mathbf{x}_j^{k+3} = a \cdot \mathbf{g}_j^k + \cdot \beta(t) \cdot \mathbf{v}_j^{k+3} \quad (26)$$

The points $\mathbf{x}_j^{k+3}$ are displayed and based on them the new value of the function L_{t_j} is calculated by setting the condition $L_{t_j}^{\,k+2} < L_{t_j}^{\,k+3} \leq L_{t_i}$. The algorithm continues analogously and stops when the value of $L_{t_j}^{\,k+n}$ satisfies the condition $L_{t_j}^{\,k+n} \geq L_{t_i}$.

Flowchart of the MH-PSO algorithm is shown in Fig. 4. The following gives a relatively complete presentation of the algorithm. The code program was write in C++ language with Visual Studio 2019 software.

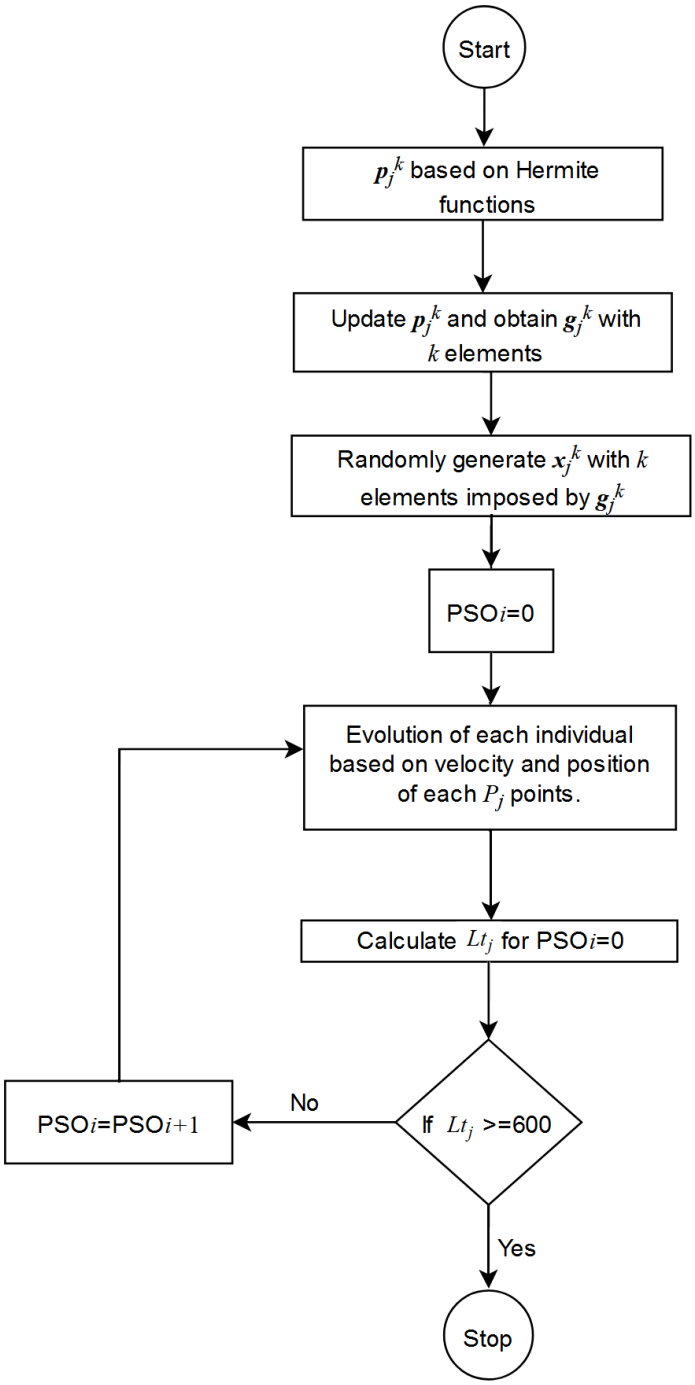

Fig. 4 Flowchart of the MH-PSO algorithm.

Algorithm code of MH-PSO

1. Objective function: $L_t = \sum s_{j+1,j}$; $s_{j+1,j}\sqrt{\left(x_{P_{j+1}} - x_{P_j}\right)^2 + \left(y_{P_{j+1}} - y_{P_j}\right)^2 + \left(z_{P_{j+1}} - z_{P_j}\right)^2}$;
2. We are generating the matrix with $\mathbf{p}_j^k$ points based on Hermite equation that contained variable $\lambda \in [0,1]$. We initialize λ with 0 and set its increment to be 0.1.
3. Further all points $\mathbf{p}_j^k$, must be validated by the velocity verification (their velocity doesn't surpass 450 mm/s). Each of these generated points represent $\mathbf{g}_j^k$.

4. We generate the matrix which contains $\mathbf{x}_j^k$, random points belonging to technological workspace with $x_{P_j} \in [-150, 150]$, $y_{P_j} \in [-150, 150]$, $z_{P_j} \in [-494, -308]$. The points are uniformly randomized and their coordinates are ascending for each x, y, z value.
5. We begin the PSO optimization imposed $L_{t_i} = 600\ mm$.
6. We initialize L_{t_i} with 0. While $L_{t_i} \leq 600$, the algorithm repeats the number of iterations increasing by one. At iteration 0, 1 and 2 (first three cases) we are determining, based on PSO equations (21,22), the velocity and, then, position coordinates of the new points.
7. After the first three iterations we can observe a recursive algorithm which we will call "the 1020 rule". The velocity of each new point that we want to generate is based on the following scheme which visually simplifies the optimization equation.

– For $\text{PSO}_0 \rightarrow \mathbf{v}_j^{k+1} = \mathbf{v}_j^k + \left(\mathbf{p}_j^k - \mathbf{x}_j^k\right) + \left(\mathbf{g}_j^k - \mathbf{x}_j^k\right)$

– For $\text{PSO}_1 \rightarrow \mathbf{v}_j^{k+2} = \mathbf{v}_j^{k+1} + \left(\mathbf{g}_j^k - \mathbf{p}_j^k\right) + \left(\mathbf{x}_0 - \mathbf{p}_j^k\right)$. For more simplicity we denote in the program code $\mathbf{x}_j^{k+1} = \mathbf{x}_0$ from PSO_0.

– For $\text{PSO}_2 \rightarrow \mathbf{v}_j^{k+3} = \mathbf{v}_j^{k+2} + \left(\mathbf{x}_0 - \mathbf{g}_j^k\right) + \left(\mathbf{x}_1 - \mathbf{g}_j^k\right)$. For more simplicity we denote in the program code $\mathbf{x}_j^{k+2} = \mathbf{x}_1$ from PSO_1.

– For $\text{PSO}_3 \rightarrow \mathbf{v}_j^{k+4} = \mathbf{v}_j^{k+3} + (\mathbf{x}_1 - \mathbf{x}_0) + (\mathbf{x}_2 - \mathbf{x}_0)$. For more simplicity we denote in the program code $\mathbf{x}_j^{k+3} = \mathbf{x}_2$ from PSO_2.

– We obtain "the 1020 rule". The next set of x_i we obtain by increasing each i by 1. The next PSO will be:

– For $\text{PSO}_4 \rightarrow \mathbf{v}_j^{k+5} = \mathbf{v}_j^{k+4} + (\mathbf{x}_2 - \mathbf{x}_1) + (\mathbf{x}_3 - \mathbf{x}_1)$.

8. The PSO ending when $L_{t_j}^{k+1} \geq L_{t_i}$.

In order to get an unbiased comparison of CPU times, all the experiments are performed using the same PC with the following configuration shown in Table 1.

Table 1. The detailed CPU settings

Hardware	
CPU	i7-7700HQ
Frequency	3.80 GHz
RAM	8 GB
Hard drive	512 GB
Software	
Operating system	Windows 10
Language	C++

The modified MH-PSO algorithm identifies those paths that are in the vicinity of the curve described by the sequence of P_j points based on the Hermite functions and that satisfy the imposed L_{t_i} length condition. The user can thus view this route and can choose from the multitude of solutions that satisfy the requirements of a certain application.

For a better visualization of the trajectory and to validate the results obtained based on the presented algorithm, we represented the sequence of P_j points inside the technological workspace for a few L_{t_j} objective function based on a CAD program. The figures above show the path generated by the sequence of points $\mathbf{p}_j^k$ (Fig. 5) and the optimized trajectory based on the modified MH-PSO algorithm for PSO_{915} and PSO_{520} (Fig. 6). We specify that all curves were drawn with the AutoLisp script program in AutoCAD software.

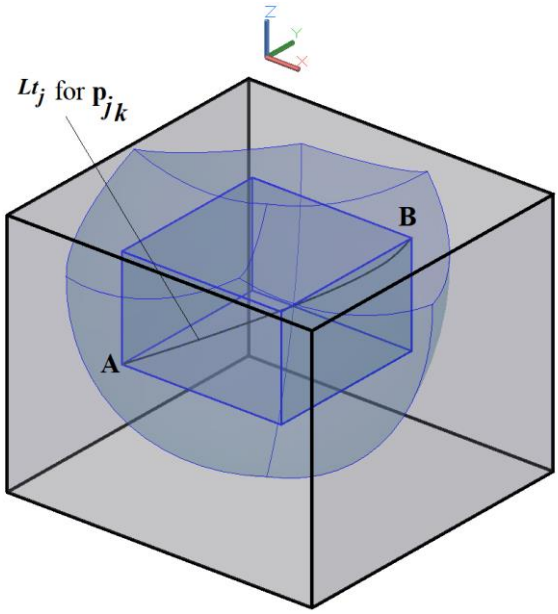

Fig. 5 The trajectory generated by $\mathbf{p}_j^k$ points.

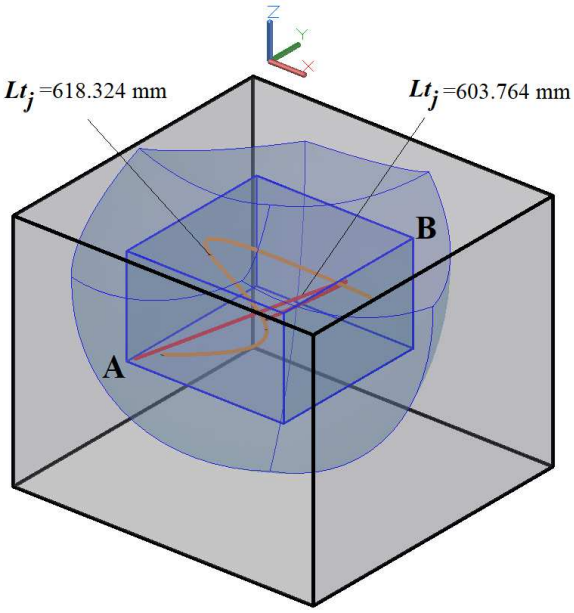

Fig. 6 The trajectory generated by MH-PSO for PSO_{915} and PSO_{520}

To be able to verify the efficiency of the algorithm, 9 Run_j sets were done.

The best value for the objective function, resulting from the algorithm for the first such set, was obtained for Run_1 as seen in the Figure 7 with $L_{t_j} = 600.85\ mm$. The highest value for the objective function is given by Run_8 with $L_{t_j} = 650.17\ mm$ (Fig. 8). Also, for the first set of 9 Run_j, we obtain a quick convergence for Run_8 after 223 iterations and a slow convergence for Run_5 and Run_9 with 957 respectively 1108 iterations (Fig. 9,10).

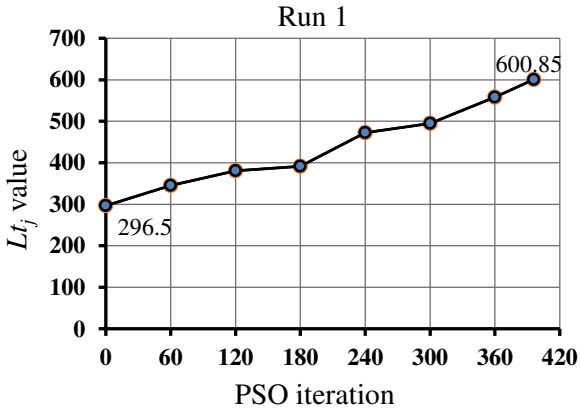

Fig. 7 Function values versus number of iterations for Run_1

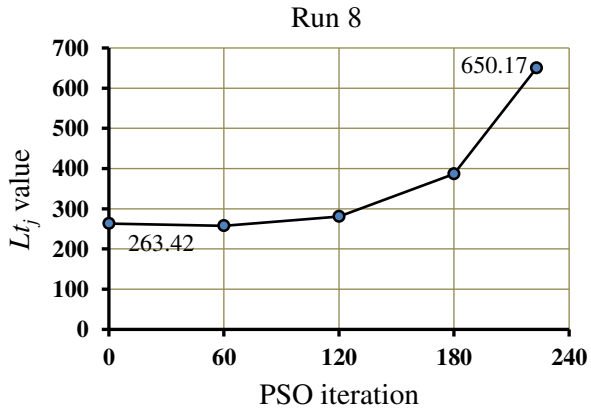

Fig. 8 Function values versus number of iterations for Run_8

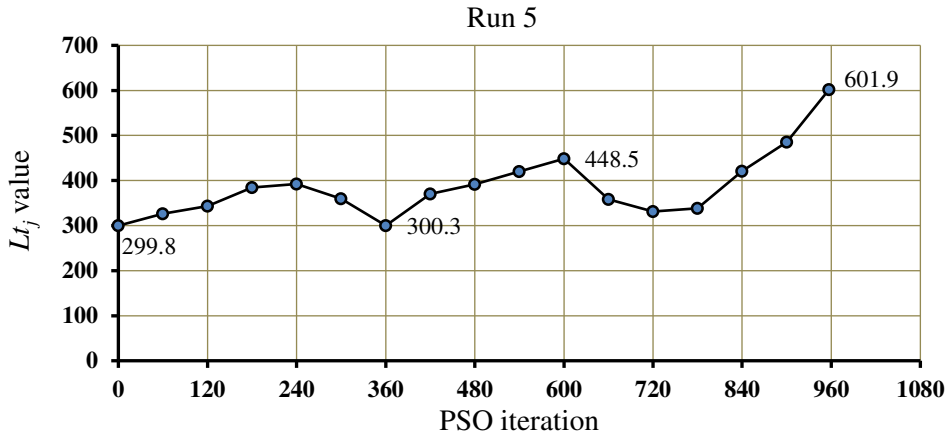

Fig. 9 Function values versus number of iterations for Run_5

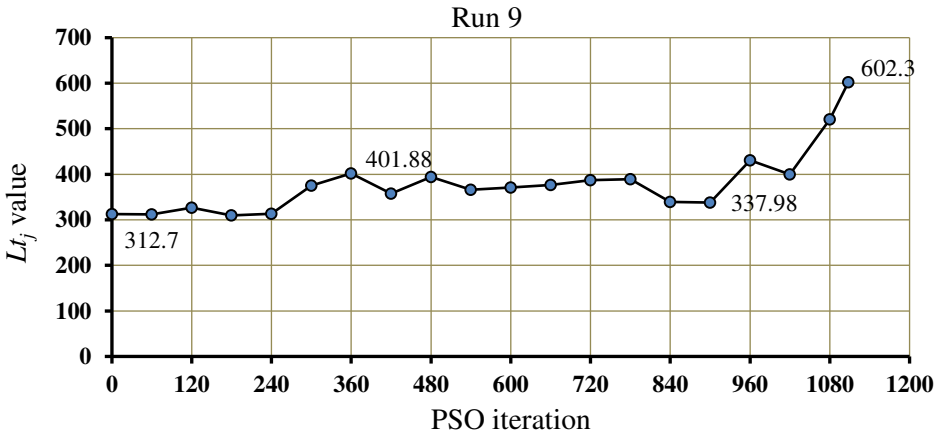

Fig. 10 Function values versus number of iterations for Run_9

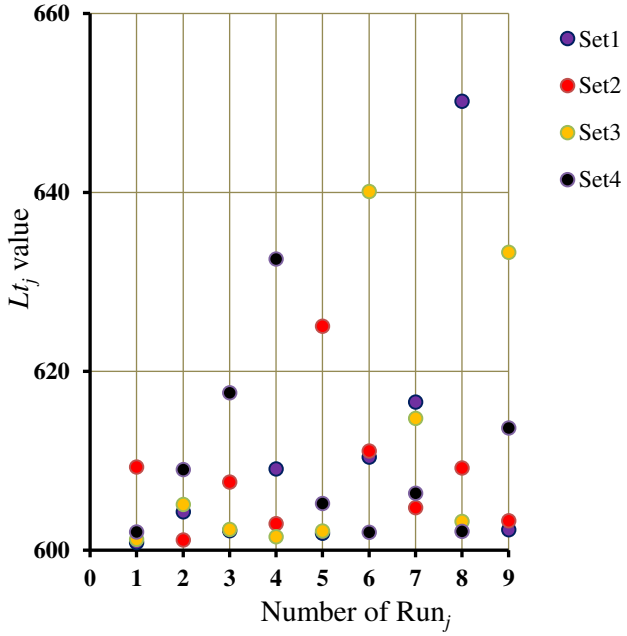

Fig. 11 Best function value for all Set_j (j=1…4)

For a better evaluation of the algorithm, we create four Set_j of nine Run_j iteration and we compared all best function values for this. The highest value for the objective function is given by Set_1 with $L_{t_j} = 650.17\ mm$ and a good dispersion of value function is obtain for Set_2 (Fig. 11). In Table 2 was listeted best, worst and mean L_{t_j} value for each Set_j number.

Table 2. Best and worst L_{t_j} value for each Set_j number

Set_j number	Best L_{t_j} value	Mean $\text{Run}_1 \dots \text{Run}_9$	Worst L_{t_j} value
1	600.85	627.22	650.17
2	601.13	613.08	625.03
3	601.22	621.13	640.08
4	602.05	615.84	632.55

Depending on the random matrices $\mathbf{R}_{1,2}^k$ and the matrix used for generated $\mathbf{x}_j^k$, the number of iterations for MH-PSO algorithm vary from PSO_{223} to PSO_{1108}. Also the time for algorithm execution vary from 1 to 4.2 seconds. In the figures below (Fig. 12) there was obtained an evolution of L_{t_j} depending on the number of PSO_j iterations for each Run_j compilation of the algorithm program.

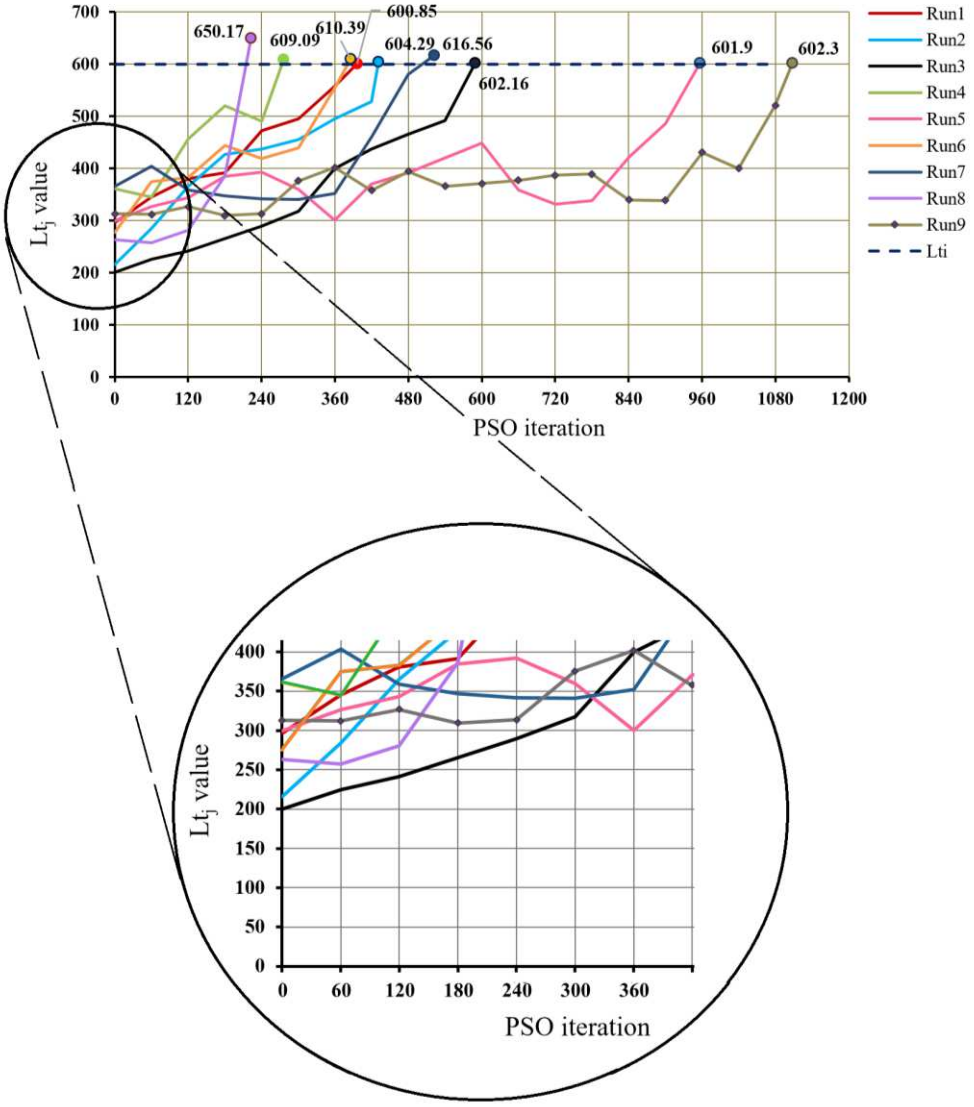

Fig. 12 L_{t_j} evolution depending on PSO iteration

4. Conclusions

The application of the PSO algorithm on a HEXA mechanism was done taking into account the complexity of the mechanisms with six degrees of freedom and the need to reduce the computational effort necessary to determine the trajectories of the positions of the characteristic point attached to the end-effector.

This paper presents a method for determining the optimal trajectory of the characteristic point based on the kinematic analysis of a parallel HEXA mechanism. The movement of the end-effector element described in the presented optimization process has a strong dynamic character. The optimization was performed based on a modified PSO algorithm ussing a Hermite polynomials (MH-PSO). During the optimization process, the function followed was the length of the trajectory described by the sequence of positions of the characteristic point, belonging to the effector element, in compliance with additional conditions imposed. The elements of the classical optimization method were highlighted, based on an objective function and some restrictions. The paper highlights that using the MH-PSO algorithm, leads to minimal effort for the analysis and mathematical formulation of the optimization problem. Addressing the optimization problem proposed in this paper provides easy implementation, stable convergence features, and good computational efficiency. The proposed optimization method can be generalized to various types of mechanisms. The approach and steps taken can be the source of inspiration for other similar issues. Indeed, the obtained results revealed that the proposed method yielded satisfactory results. To validate theese results obtained based on the presented algorithm, the sequence of P_j points inside the technological workspace was represented for a few L_{t_j} objective functions using a script program implemented in a CAD software.

Bibliography

1. Lee KY and El-Sharkawi MA. Modern Heuristic Optimization Techniques Theory and Applications to Power Systems. ed. Wiley- IEEE Press, 2008, pp. 173.
2. Erdogmus P and Toz M. Heuristic Optimization Algorithms in Robotics. In: Küçük S (eds) Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization. ed. INTECH Open Access Publisher, 2012, pp. 315-316.
3. Lazinca A. Particle Swarm Optimization. ed. InTech, 2009, pp. 10-13.
4. Mo H and Xu L. Research of biogeography particle swarm optimization for robot path planning. J. Neurocomputing 2015; 148: 91-99.
5. Lee KB and Kim JH. Multiobjective Particle Swarm Optimization With Preference-Based Sort and Its Application to Path Following Footstep Optimization for Humanoid Robots. J. IEEE Trans. Evol. Comput. 2013; 17: 755-766.
6. Wang F, Wang Z, Lin M. Robot Path Planning Based on Improved Particle Swarm Optimization. In: ICBAIE 2021, Nanchang, China, 26-28 March 2021, pp. 887 – 891.
7. Sheng XN, Xie W. Real-time path planning for mobile robots based on quantum evolutionary algorithm. J. Comput. Sci. 2013; 40: 229-232.
8. Kennedy J, Eberhart R. Particle Swarm Optimization. J. Proceedings of IEEE INNC 1995; 4: 1942–1948.
9. Li J and Xiao X. Multi- Swarm and Multi- best particle swarm optimization algorithm. In: Proceeding of the 7th WCICA 2008, Chongqing, China, 25-27 June 2008, pp. 6281-6286.
10. Zargar AN and Javadi H. Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target. In: 3th UKSIM/AMSS EMS, Athens, Greece, 25-27 November 2009, pp. 60-65.
11. Gueaieb W and Miah MS. Mobile robot navigation using particle swarm optimization and noisy RFID communication. In: CIMSA 2008, Istanbul, Turkey, 14-16 July 2008, pp. 111-116.
12. Xu W, Li C, Liang B, Liu Y, Xu Y. The Cartesian path planning of free-floating space robot using particle swarm optimization. Int J Adv Robot Syst. 2008; 5(3): 27.

13. Hao Y, Zu W, Zhao Y. Real-time obstacle avoidance method based on polar coordination particle swarm optimization in dynamic environment. In: 2007 2nd IEEE conference on industrial electronics and applications, 2007; pp 1612–1617.
14. Xu Z, Li S, Chen Q, Hou B. MOPSO based multi-objective trajectory planning for robot manipulators. In: 2015 2nd international conference on information science and control engineering. IEEE, 2015; pp 824–828.
15. Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. J. IEEE Trans. Evol. Comput. 2002; 6: 58–73.
16. Eberhart RC and Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the IEEE CEC 2000, La Jolla, CA, USA, 16-19 July 2000, pp.84–88.
17. Trelea IC. The particle swarm optimization algorithm: convergence analysis and parameter selection, J. Inf. Process. Lett. 2003; 85: 317–325.
18. Jiang M, Luo Y and Yang S. Particle swarm optimization-stochastic trajectory analysis and parameter selection. In: F. Chan and M. Kumar Tiwari (eds), Swarm intelligence. Focus on ant and particle swarm optimization, ed. I-TECH Education and Publishing, 2007, pp.179–198.
19. Liu Q. Order-2 stability analysis of particle swarm optimization. J. EVCO. 2015; 23(2): 187–216.

Declaration

Availability of data and materials section

-All data generated or analysed during this study are included in this published article.

Competing interests

-The authors declares that they have no competing interests.

Funding

-The research was funded by the Ministry of National Education and Scientific Research from Romania.

Authors' contributions

-Each author has made substantial contributions to the conception and design of the work, analysis and interpretation of data. Also the creation of new alghoritm used in the work and drafted the work was made by both authors. The work was revised it by the first author.

Both authors approved the submitted version (and any substantially modified version that involves the author's contribution to the study).

Both authors have agreed to be personally accountable for the contributions and the accuracy or integrity of any part of the work.

Acknowledgements

-The research was funded by the Ministry of National Education and Scientific Research from Romania.