Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 9815469, 9 pages
https://doi.org/10.1155/2018/9815469

Research Article

A Hybrid Strategy of Differential Evolution and Modified Particle Swarm Optimization for Numerical Solution of a Parallel Manipulator

Bingyan Mao,[1,2] **Zhijiang Xie**,[1] **Yongbo Wang**,[2] **Heikki Handroos**,[2] **and Huapeng Wu**[2]

[1] *State Key Laboratory of Mechanical Transmission, Chongqing University, Chongqing 400044, China*
[2] *Lappeenranta University of Technology, 53810 Lappeenranta, Finland*

Correspondence should be addressed to Bingyan Mao; sucaogen@163.com

Received 19 July 2017; Revised 8 January 2018; Accepted 22 January 2018; Published 22 February 2018

Academic Editor: Giuseppe Fedele

This paper presents a hybrid strategy combined with a differential evolution (DE) algorithm and a modified particle swarm optimization (PSO), denominated as DEMPSO, to solve the nonlinear model of the forward kinematics. The proposed DEMPSO takes the best advantage of the convergence rate of MPSO and the global optimization of DE. A comparison study between the DEMPSO and the other optimization algorithms such as the DE algorithm, PSO algorithm, and MPSO algorithm is performed to obtain the numerical solution of the forward kinematics of a 3-RPS parallel manipulator. The forward kinematic model of the 3-RPS parallel manipulator has been developed and it is essentially a nonlinear algebraic equation which is dependent on the structure of the mechanism. A constraint equation based on the assembly relationship is utilized to express the position and orientation of the manipulator. Five configurations with different positions and orientations are used as an example to illustrate the effectiveness of the proposed DEMPSO for solving the kinematic problem of parallel manipulators. And the comparison study results of DEMPSO and the other optimization algorithms also show that DEMPSO can provide a better performance regarding the convergence rate and global searching properties.

1. Introduction

Differential evolution (DE) is a heuristic and straightforward strategy with the prominent features of global optimization and only a few control parameters [1, 2]. Particle swarm optimization (PSO) is a group theory based algorithm that was inspired by fish schools, bird flocks, and others. The disadvantage of PSO is that the individuals are easily influenced by the best particle and best position so it may only get the local optimum [3, 4]. The control parameters of PSO can be modified according to the specific optimization problems and applications [5–7]. This paper presents a new method integrated with differential evolution (DE) and modified particle swarm optimization (PSO). In particular, this strategy aims to combine the advantages of DE and the modified PSO together and then apply the hybrid algorithm to the numerical solution of a parallel manipulator.

Parallel robots have been a hot research topic for many years due to their superior performance such as high response speed, high stiffness, high payload/weight ratio, low inertia, and good dynamic performance [8–10]. This paper focuses on a spatial parallel manipulator with 3 DOFs which was developed by Lee and Shah [11]. The 3-RPS parallel manipulator has three legs and each branch is a serial kinematic chain [12]. To obtain the position and orientation of the moving platform, it is necessary to define the forward kinematics of the parallel manipulator based on the length of the branches [13]. It should be noted that forward kinematics of parallel manipulators is more complicated than that of serial ones, and vice versa [14–16].

Analysis of kinematics can be divided into two approaches: analytical methods and numerical methods [17, 18]. In numerical methods, the forward kinematic solution is found by solving a nonlinear global optimization problem to find

the numerical solution [19]. Many numerical calculation methods for solving nonlinear equations have been utilized in kinematic problems. Newton's iteration method, together with its improvements, is a common method that is very efficient as regards convergence speed. However, Newton's iteration method is an arduous procedure which is sensitive to initial values and involves a large number of calculation steps [8]. Compared to PSO, the differential evolution (DE) algorithm has received much more attention due to its capability of global optimization [20, 21].

The nonlinearity and multiple solution properties of the forward kinematics of a parallel manipulator make the analytical method more difficult and sophisticated than the numerical one. The numerical solution of the forward kinematics can be obtained using optimization methods such as DE and PSO or other hybrid optimization methods [22, 23]. In this paper, some modern intelligent optimization methods will be utilized and compared with each other. A time-saving hybrid strategy combined by differential evolution and modified particle swarm optimization is developed for the numerical solution of the forward kinematics of a 3-DOF parallel manipulator.

2. A Hybrid Strategy of Differential Evolution and Particle Swarm Optimization

The differential evolution (DE) optimization algorithm is a simulation of the biological evolution process. It is capable of handling nondifferentiable, nonlinear, and multimodal objective functions. To start an optimization process, an initial population must be randomly generated within a predefined bound, and then a new population in the next generation is generated through mutation, crossover, and selection operations.

Particle swarm optimization (PSO) is a computational algorithm that optimizes a problem by iteratively improving a candidate solution with regard to a given measure requirement. The movements of the particles mimic the movement of organisms in a bird flock or fish school and are guided by their own best known position in the search space as well as the entire swarm's best known position.

Modified particle swarm optimization (MPSO) has the user-defined constant parameters ω, c_1, and c_2 which have a great impact on the optimization performance. The parameter ω is utilized to adjust the velocity, the parameter c_1 is utilized to adjust $p\text{best}(t)$, which is the best position achieved so far by every individual, and the parameter c_2 is utilized to adjust $g\text{best}(t)$, which is the best value obtained so far by any particle in the population. In order to improve the performance of PSO, time-varying acceleration coefficients and time-varying inertia weight can be utilized. The inertia weight can provide a balance between the local search and global search during the optimization process:

$$\omega = (\omega_1 - \omega_2) \times \frac{(\text{Iter}_{\max} - \text{Iter})}{\text{Iter}_{\max}} + \omega_2, \tag{1}$$

where ω_1 and ω_2 are the initial and the final inertia weight, respectively, Iter is the current iteration number, and $\text{Iter}_{\max}$ is the maximum number of iterations.

By increasing the number of iterations, the weights of $p\text{best}(t)$ and $g\text{best}(t)$ can have different convergence rates:

$$\begin{aligned} c_1 &= (c_{1\min} - c_{1\max}) \times \frac{\text{Iter}}{\text{Iter}_{\max}} + c_{1\max} \\ c_2 &= (c_{2\max} - c_{2\min}) \times \frac{\text{Iter}}{\text{Iter}_{\max}} + c_{2\min}, \end{aligned} \tag{2}$$

where $c_{\max}$ and $c_{\min}$ are the maximum and minimum acceleration coefficient of c_1 and c_2, respectively.

The hybrid strategy of DE and MPSO (DEMPSO) is the new strategy. A key merit of DE algorithm is the efficient global optimization. Furthermore, the diversity of the entire population can be always maintained during the whole evolution process, which can prevent the individuals from falling into a local optimum. PSO, on the other hand, has the advantage of fast convergence speed. The individual with the best history and the best individual among the entire iteration are saved to obtain the lowest fitness values.

Combining the advantages of DE and PSO, a new hybrid DEMPSO strategy is proposed which aims to achieve both fast convergence speed and efficient global optimization. Since the population of PSO easily falls into a local optimum, the proposed DEMPSO method uses the DE algorithm to reduce the search space first, and then the obtained populations are taken over by the MPSO as an initial population to get a fast convergence rate to the final global optimum. The hybrid algorithm can obtain the global minimum value based on the fitness function, which is built for the numerical solution for forward kinematics of a parallel manipulator. The procedure of the DEMPSO algorithm is illustrated as follows.

(1) Population Initialization. The individual x, with the population number NP, is randomly generated to form an initial population in a D-dimensional space. All the individuals should be generated within the bounds of the solution space. The initial individuals are generated randomly in the range of the search space. And the associated velocities of all particles in the population are generated randomly in the D-dimension space. Therefore, the initial individuals and the initial velocity can be expressed as follows:

$$\begin{aligned} X_i(0) &= \{x_{i,1}(0), x_{i,2}(0), \ldots, x_{i,D}(0)\} \\ x_{i,j}(0) &= x_{\min} + \text{rand}_{i,j}(0,1) \times (x_{\max} - x_{\min}) \\ V_i(0) &= \{v_{i,1}(0), v_{i,2}(0), \ldots, v_{i,\text{Np}}(0)\}, \end{aligned} \tag{3}$$

where $1 \le i \le \text{Np}$, $1 \le j \le D$, $[x_{\min}, x_{\max}]$ is the range of the search space, and rand$(0, 1)$ is a random number chosen between 0 and 1.

(2) Iteration Loop of DE. Let individual $X_i(t)$ denote the mutation operation at time t. By randomly choosing three

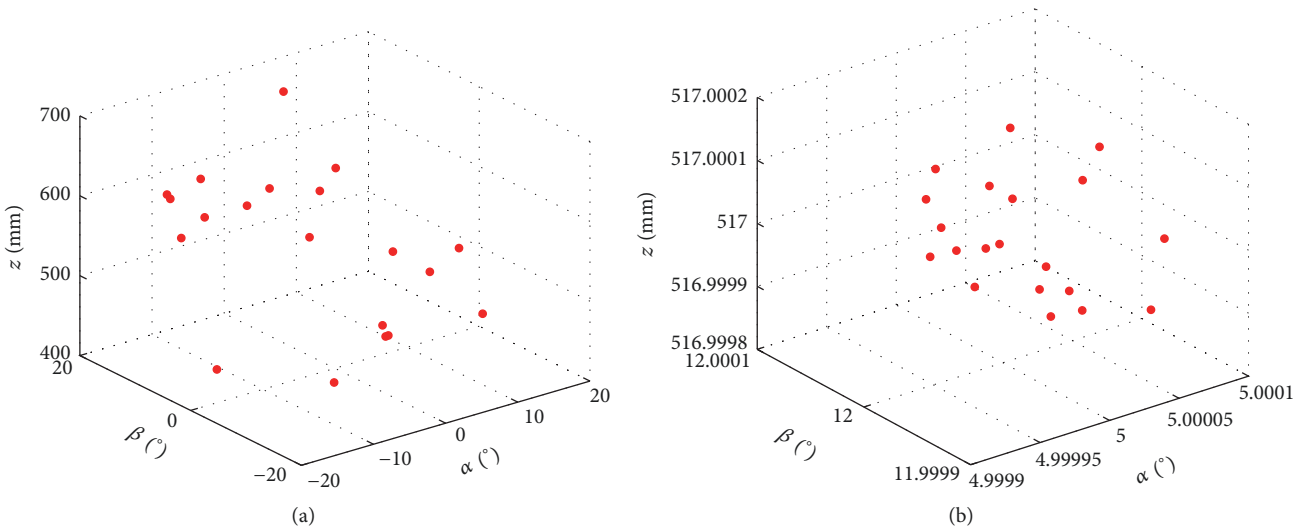

FIGURE 1: Position of the population of DE. (a) At the beginning of the simulation; (b) on convergence.

individuals from the previous population, the mutant individual $V_i(t+1)$ can be generated as the following equation:

$$V_i(t+1) = X_{r1}(t) + F\left(X_{r2}(t) - X_{r3}(t)\right), \tag{4}$$

where F is a differential weight between 0 and 1. It is a zoom factor to control the amplification of the mutation operation. r_1, r_2, and r_3 are random integers that have been selected from 1 to NP and r_1, r_2, r_3, and i are not the same as each other.

The crossover operation aims to construct a new population $u_{i,j}(t+1)$ which is chosen from the current individuals and mutant individuals in order to increase the diversity of the generated individuals:

$$\begin{aligned} &U_i(t+1) = \left\{u_{i,1}(t+1), u_{i,2}(t+1), \ldots, u_{i,D}(t+1)\right\} \\ &u_{i,j}(t+1) \\ &\quad = \begin{cases} v_{i,j}(t+1), & \text{if rand}(0,1) \le \text{Cr or } j = j_{\text{rand}} \\ x_{i,j}(t), & \text{otherwise,} \end{cases} \end{aligned} \tag{5}$$

where rand(0, 1) is a random number chosen from 0 to 1, j_{rand} is an integer chosen from 1 to D randomly, and Cr is a crossover parameter that is randomly chosen from 0 to 1.

In the selection operation, the crossover vector $U_i(t+1)$ is compared to the target vector $X_i(t)$ by evaluating the fitness function value based on a greedy criterion, and the vector with a smaller fitness value is selected as the next generation vector:

$$\begin{aligned} &X_i(t+1) \\ &\quad = \begin{cases} U_i(t+1), & \text{if } f\left(X_i(t)\right) \ge f\left(U_i(t+1)\right) \\ X_i(t), & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

Update the global best part with the minimum fitness value ($g$best) and the personal-best part ($p$best). Let the $g$best value be F_1 and perform a comparison with the stopping tolerance value E_1. If F_1 is less than E_1, the iteration of DE has finished. All the population and positions will continue to the next loop of MPSO.

(3) Iteration Loop of MPSO. Set the time-varying parameters c_1, c_2, and ω as the MPSO defined. Renew the individual velocity as follows:

$$\begin{aligned} v_{i,j}(t+1) = {} & w \cdot v_{i,j}(t) + c_1 \cdot \text{rand}(0,1) \\ & \cdot \left(p\text{best}_{i,j}(t) - x_{i,j}(t)\right) + c_2 \cdot \text{rand}(0,1) \\ & \cdot \left(g\text{best}_{i,j}(t) - x_{i,j}(t)\right), \end{aligned} \tag{7}$$

where the constant parameters ω, c_1, and c_2 are defined and rand(0, 1) is a number randomly chosen between 0 and 1.

The new individuals are generated as follows:

$$x_{i,j}(t+1) = x_{i,j}(t) + v_{i,j}(t+1). \tag{8}$$

Let the $g$best value of MPSO be F_2 and perform a comparison with fitness value E_2. If F_2 is less than E_2, the iteration of MPSO has finished.

The initial population is generated by the DE algorithm, and the stopping criterion of DE is set as the fitness value F_1 less than a user-defined stopping tolerance value E_1 which is dependent on the specific kinematics of a parallel manipulator. When the fitness value F_1 is less than E_1, the DE population will be taken over by the MPSO algorithm. The new velocity and new location of the population are updated in every generation until the fitness value F_2 becomes less than the bound condition E_2 satisfied.

In order to depict clearly the population moving process, the position and difference vector distribution of the population for the DE-based algorithm and MPSO-based algorithm are shown in Figures 1–4.

The initial and final distributions of the DE-based population are shown in Figures 1(a) and 1(b), respectively. The position of individuals of the DE-based algorithm has

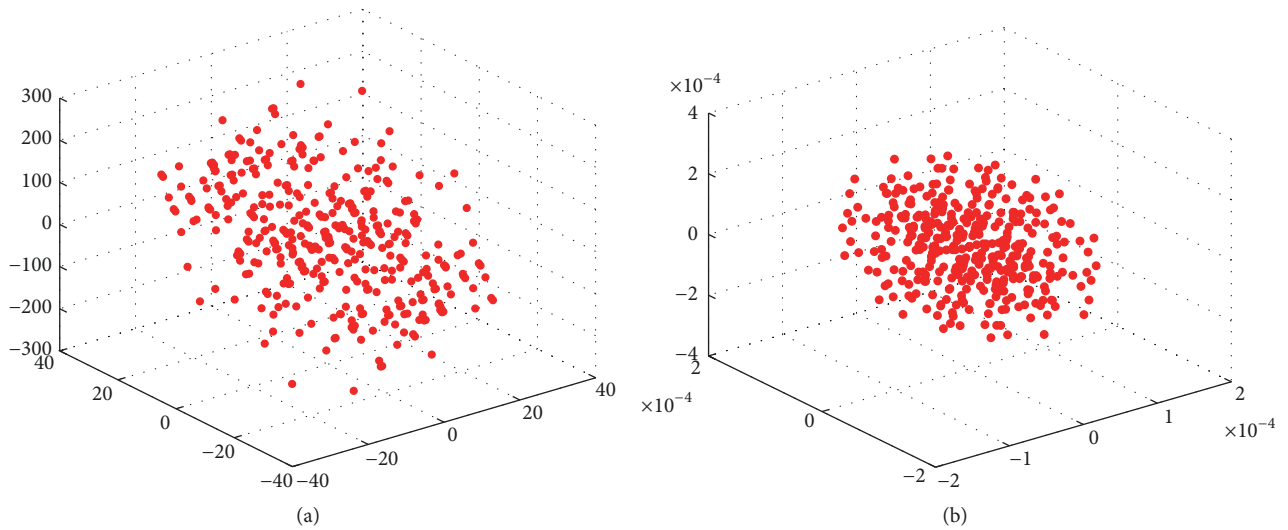

Figure 2: Difference vector distribution of DE. (a) At the beginning of the simulation; (b) on convergence.

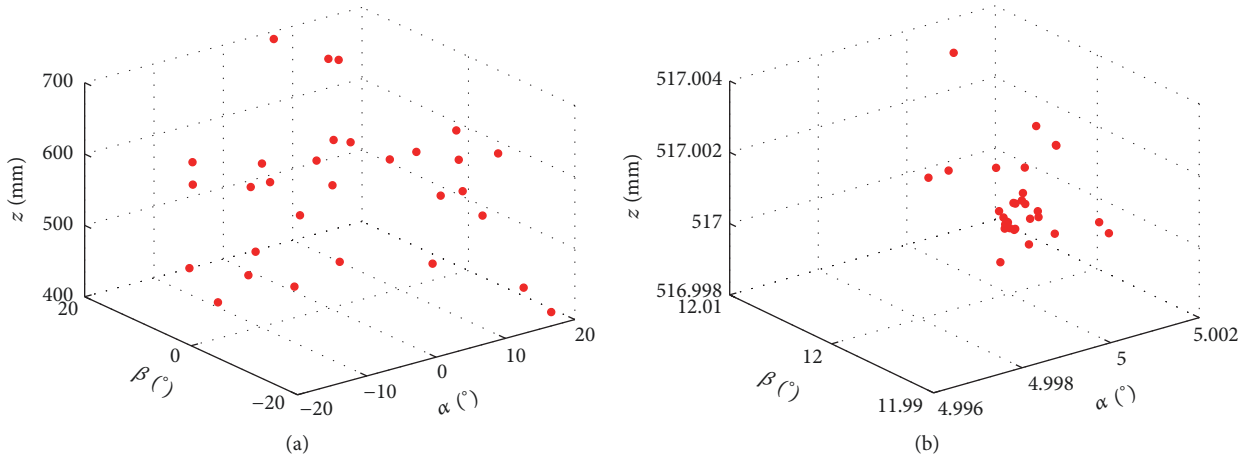

Figure 3: Position of the population of MPSO. (a) At the beginning of the simulation; (b) on convergence.

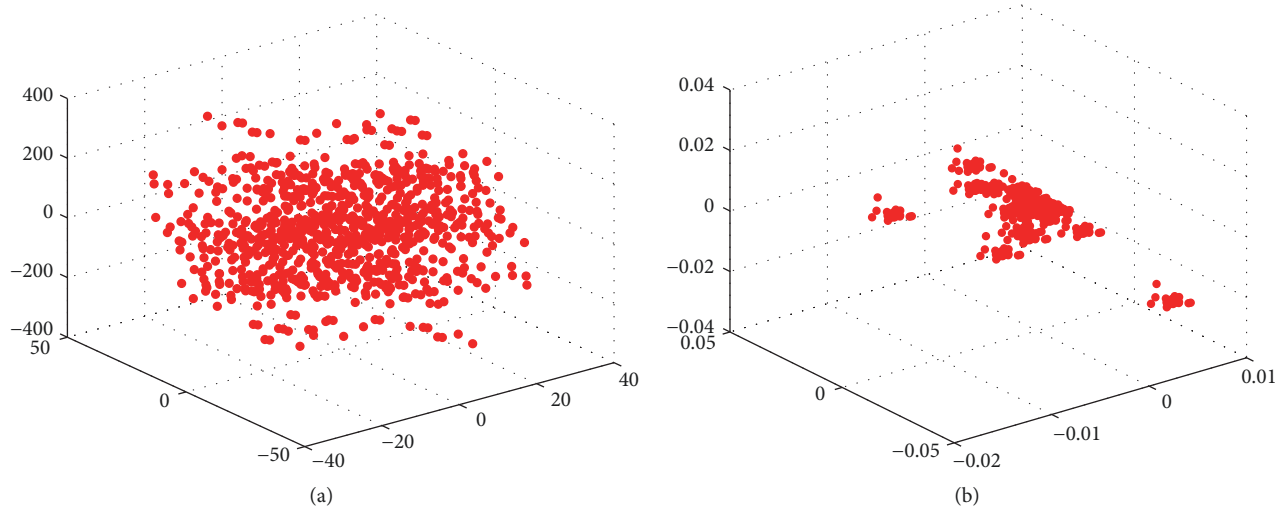

Figure 4: Difference vector distribution of MPSO. (a) At the beginning of the simulation; (b) on convergence.

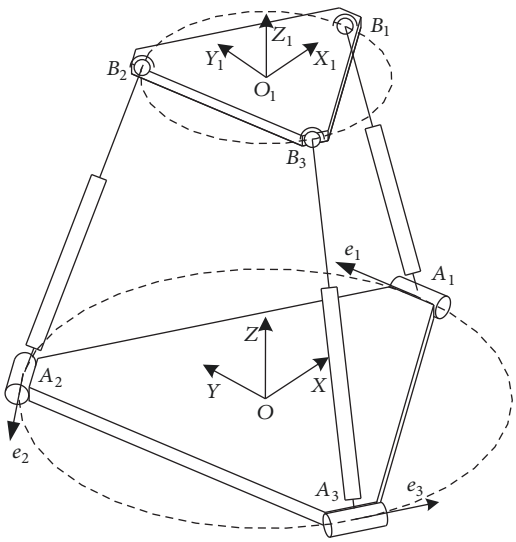

Figure 5: A 3-RPS parallel manipulator.

converged to 10^{-4}. The initial and final distributions of the MPSO-based population are shown in Figures 3(a) and 3(b), respectively. The position of individuals of the MPSO-based algorithm has converged to 10^{-3}.

The individuals of the MPSO-based algorithm are not gathered in a small space like the individuals of the DE-based algorithm. Based on the fitness value, the best individual of the MPSO-based algorithm will be chosen at every generation, and the algorithm will converge when the best individual meets the fitness value set before.

The initial and final difference vector distributions of the DE-based algorithm are shown in Figures 2(a) and 2(b), respectively. The initial difference vector distribution of the MPSO-based algorithm shown in Figure 4(a) is similar to that of the DE-based algorithm. Comparison of the final difference vector of the MPSO-based algorithm, shown in Figure 4(b), with that of the DE-based algorithm, shown in Figure 2(b), indicates that the population of the MPSO-based algorithm stops the iteration without convergence of all the individuals, while the individuals of the DE-based algorithm converge to a small space as the global optimization.

3. Forward Kinematics of a Parallel Manipulator

A parallel manipulator with 3 DOFs was proposed by Fang and Huang [12], which has been widely used in airplane simulators, walking machines, and others. The 3-RPS parallel robot is composed of a base platform, three legs, and a moving platform, as shown in Figure 5. Each leg is a serial chain consisting of a revolute (r) joint, a prismatic (p) joint, and a spherical (s) joint. The manipulator has three degrees of freedom: two rotations are about the x-axis and y-axis and one translation is along the z-axis. The p joints are driven by linear actuators so that the moving platform can achieve the required position and orientation.

The global coordinate O-xyz is built at the center of the base platform. The orientation of the x-axis is from point O to point A_1, and the orientation of the y-axis is parallel to the line $\overrightarrow{A_2A_3}$. The r joints are evenly distributed around the base platform as an equilateral triangle. The moving coordinate O_1-$x_1y_1z_1$ is built at the center of the moving platform. The orientation of the x_1-axis is from point O_1 to point B_1, and the orientation of the y_1-axis is parallel to the line $\overrightarrow{B_2B_3}$. The s joints are distributed as r joints. For simplicity and without loss of generality, all coordination systems abide by the right-hand rule. The geometric parameters of the manipulator are the radius r_a of the base platform and the radius r_b of the moving platform.

Let R and P denote the rotation matrix and position vector which move from the global coordinate system to the moving coordinates, respectively. Then, the typical kinematic chain can be denoted as a mathematical formula:

$$\overrightarrow{A_iB_i} = R\overrightarrow{O_1B_i} + P - \overrightarrow{OA_i} \quad i = 1, 2, 3, \tag{9}$$

where $\overrightarrow{A_iB_i}$ is a vector from point A_i to point B_i. The vectors $\overrightarrow{O_1B_i}$ and $\overrightarrow{OA_i}$ belong to the moving coordinates and the global coordinates, respectively. R is Z-Y-X Euler rotation transformation matrix and P is position transformation vector.

$$R = \begin{bmatrix} c\gamma c\beta & c\gamma s\beta s\alpha - s\gamma c\alpha & c\gamma s\beta c\alpha + s\gamma s\alpha \\ s\gamma c\beta & s\gamma s\beta s\alpha + c\gamma c\alpha & s\gamma s\beta c\alpha - c\gamma s\alpha \\ -s\beta & c\beta s\alpha & c\beta c\alpha \end{bmatrix},$$
$$P = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix}, \tag{10}$$

where $s\alpha = \sin\alpha$ and $c\alpha = \cos\alpha$.

The position vectors of points A_i and B_i with respect to the global coordinate system and the moving coordinates can be expressed as follows:

$$\overrightarrow{OA_i} = r_a\left(\cos\phi_i, \sin\phi_i, 0\right)^T$$
$$\overrightarrow{O_1B_i} = r_b\left(\cos\phi_i, \sin\phi_i, 0\right)^T \tag{11}$$
$$\phi_i = \frac{2\pi}{3}(i-1) \quad i = 1, 2, 3.$$

A constraint equation, based on the assembly relationship where the revolute joint axis is perpendicular to the prismatic joint axis, can be written as follows:

$$\left(R\overrightarrow{O_1B_i} + P\right)^T \cdot e_i = 0$$
$$e_i = r_a\left(\cos\varphi_i, \sin\varphi_i, 0\right)^T \quad \varphi_i = \phi_i + \frac{\pi}{2}, \tag{12}$$

where e_i is the axis of the revolute joint and its coordinate values are shown in Table 1.

Table 1: Coordinate values of each rotation axis.

	e_1	e_2	e_3
x	0	$-\sqrt{3}/2$	$\sqrt{3}/2$
y	1	$-1/2$	$-1/2$

The length of each link $\overrightarrow{A_iB_i}$, that is, the inverse kinematics of the manipulator, can be calculated as a closed chain as follows:

$$d_i = \sqrt{\left(R\overrightarrow{O_1B_i} + P - \overrightarrow{OA_i}\right)^T \left(R\overrightarrow{O_1B_i} + P - \overrightarrow{OA_i}\right)} \quad i = 1, 2, 3. \tag{13}$$

Given a set of lengths of the prismatic joints, the forward kinematics is to get the position and orientation of the moving platform. The numerical forward kinematic solutions of the 3-RPS parallel manipulator are a nonlinear algebraic equation. There are three rotations and three translations in the transformation matrix, but only two rotations and one translation are left in the forward kinematics since there are three extra constraints equations. Based on the specific structure of the parallel manipulator, the constraint equations can be obtained as follows:

$$\begin{aligned} x_e &= f(\alpha, \beta, \gamma) \\ &= \frac{r_b}{2}(\cos\gamma\cos\beta - \sin\gamma\sin\beta\sin\alpha - \cos\gamma\cos\alpha) \\ y_e &= f(\beta, \gamma) = -r_b \cdot \sin\gamma\cos\beta \\ \gamma &= f(\alpha, \beta) = \arctan\left(\frac{\sin\alpha\sin\beta}{\cos\alpha + \cos\beta}\right). \end{aligned} \tag{14}$$

Substituting (14) into (13), the inverse kinematics of (13) can be simplified as

$$d_i = f_i(\alpha, \beta, z_e) \quad i = 1, 2, 3. \tag{15}$$

For the parallel manipulator, the analytical inverse kinematic solution of the above equation is straightforward. The nominal leg length l_i can be obtained easily based on the assumed real pose α^r, β^r, and ${z_e}^r$. However, given l_i, the analytical solution for the simulated pose α^s, β^s, and ${z_e}^s$ becomes very complicated and it may have multiple solutions. The given l_i is obtained based on the real pose α^r, β^r, and ${z_e}^r$. So, the forward kinematics based on the given l_i is to let the simulated pose α^s, β^s, and ${z_e}^s$ infinitely approach the real pose α^r, β^r, and ${z_e}^r$. The simulated pose converges to the real pose with the convergence of the algorithm to the global minimum. The numerical solution of this problem can be solved by minimizing the difference between the given length l_i and the predicted length d_i calculated from (15) to find a set of optimized α^s, β^s, and ${z_e}^s$. The fitness function based on the least squares method can be written as

$$f_{ss} = \min \sum_{i=1}^{3} \left(l_i - f_i(\alpha^s, \beta^s, {z_e}^s)\right)^2, \tag{16}$$

Table 2: Workspace of the moving platform.

	Direction of x-axis	Direction of y-axis	Direction of z-axis
Range of translation (mm)	0	0	[400, 700]
Range of rotation (deg.)	[−20, 20]	[−20, 20]	0

where l_i is a known leg length which can be acquired from the linear actuator and $f_i(\alpha^s, \beta^s, {z_e}^s)$ is used to calculate the predicted leg length during the optimization process.

4. Case Simulation and Results

For the manipulator studied in this work, the task of the simulation is to obtain the end-effector pose when the length of the legs is already known. In the simulation, the geometric parameters of the manipulator are $r_a = 274$ mm and $r_b = 158$ mm. The workspace of the moving platform of the manipulator is given in Table 2.

For a specific pose selected in the workspace, for instance, $\alpha^r = 5^\circ$, $\beta^r = 12^\circ$, and ${z_e}^r = 517$ mm, through an inverse calculated as $l_1 = 499.0178$ mm, $l_2 = 557.7314$ mm, and $l_3 = 534.3032$ mm. On the contrary, if these three-leg lengths have been obtained from linear actuators, then the numerical optimization task is to search for an optimal combination of α^s, β^s, and ${z_e}^s$ to minimize the fitness function. In our first simulation, the DE-, PSO-, and MPSO-based optimization algorithms will be employed to solve this numerical problem. The aim is to investigate the performance of each algorithm and find a possible solution with the characteristics of fast convergence rate and global optimization. The simulations were implemented using Matlab R2012b on a computer with an Intel® Core™ i7-4510U CPU @ 2.00 GHz and 7.71 GB RAM. During simulation, the classical DE algorithm DE/rand/bin with the control parameters of $F = 0.85$ and Cr = 1 is chosen; the PSO control parameters are chosen as $c_1 = 2$, $c_2 = 2$, and $\omega = 0.9$. For the control parameters of MPSO, c_1 was set to gradually decrease from 2.5 to 0.5 and c_2 was set to gradually increase from 0.5 to 2.5. The inertia weight factor ω was set in two ways: (1) the inverse way (here, we called it MPSO^1, where the control parameter ω was set to gradually decrease from 0.9 to 0.6) and (2) the direct way (here, we called it MPSO^2, where the control parameter ω was set to gradually increase from 0.6 to 0.9).

Using the above parameters and given the same population number of 30 and the same stopping criterion of $1e^{-08}$, the simulation results of the DE-, PSO-, and MPSO-based algorithms with different iterations and computation times are listed in Table 3. From the simulation results, it can be seen that the computation time of the DE-based algorithm is less than of other algorithms, but it takes more iterations to converge; on the other hand, the MPSO^1 algorithm has the fastest convergence rate with only 560 iterations, but the computation time is a little greater than in the DE algorithm.

Figure 6 illustrates the logarithm-based fitness function values of DE, PSO, and MPSO with respect to the generations.

Table 3: Simulation results of forward kinematics with DE, PSO, and MPSO.

	Pop. number	Fitness value	Iteration	Time (sec)
DE	30	$1e-08$	3120	0.859791
PSO	30	$1e-08$	1137	1.608181
MPSO^1	30	$1e-08$	560	1.119108
MPSO^2	30	$1e-08$	2433	3.561454

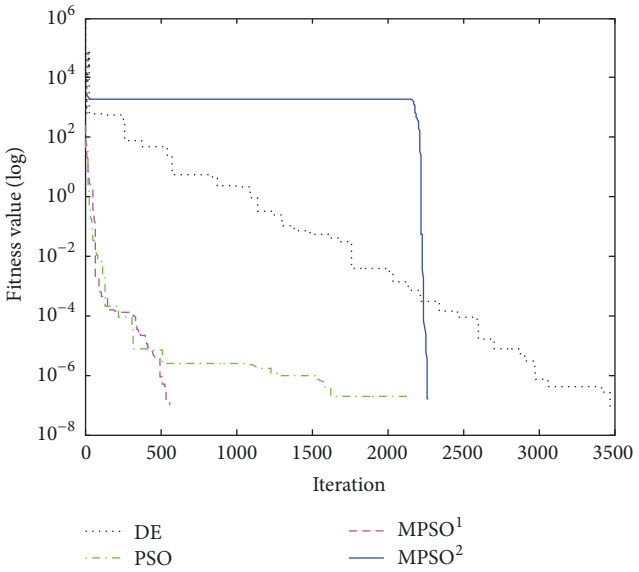

Figure 6: Logarithm of fitness function values with DE, PSO, and MPSO.

It can be seen that the convergence rate of the DE-based algorithm declines gradually. The fitness value of PSO and MPSO^1 drops very fast at the beginning, but the decline becomes gentler after a number of iterations. The fitness value of MPSO^2 has not got any changes before 2000 generations but it suddenly converged after that period; this is a very interesting characteristic which would be utilized. In our proposed hybrid method, that is, DEMPSO, the DE algorithm part is employed to bypass the steady-state part of MPSO^2, and the MPSO^2 part is used to obtain a fast convergence rate.

For the proposed DEMPSO algorithm, besides the control parameter selection, one important issue is to decide the break fitness value E_1 for the DE algorithm since this value will influence the total computation time after MPSO^2 takes over the optimization process. Table 4 presents the DEMPSO simulation results of the forward kinematics with different break fitness values of DE. For different break fitness values E_1 of DE and the same stopping fitness value E_2 of MPSO^2, the function ran 30 times to get average results of the total iterations of DE and MPSO^2, the computation time of DE at break point E_1, and the total computation time of DEMPSO at terminal point E_2. From the table, it can be seen that the optimization will spend the least amount of time when the fitness value E_1 is equal to 70.

By selecting the fitness value E_1 as a break point of DEMPSO and DEPSO, the comparison results of forward

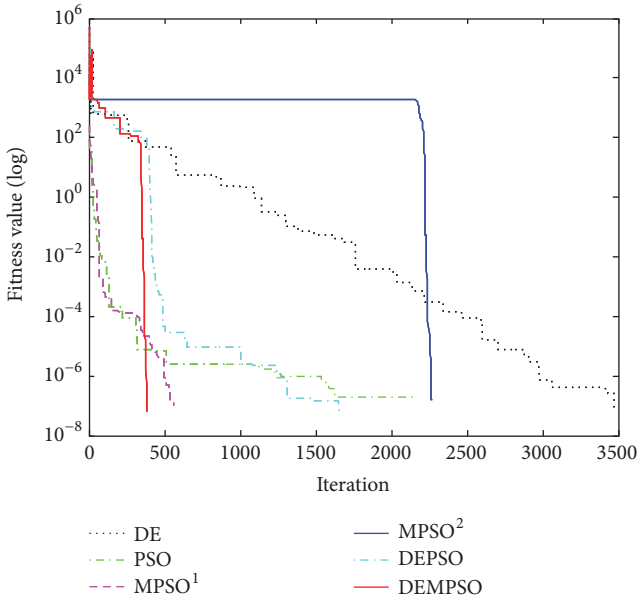

Figure 7: Logarithm-based fitness values with DE, PSO, MPSO, DEPSO, and DEMPSO.

kinematics with DEMPSO, DEPSO, DE, PSO, and MPSO are listed in Table 5. The logarithm-based fitness function values with respect to iterations are plotted in Figure 7. It can be clearly seen that DEMPSO has great advantage not only as regards convergence speed but also as regards the number of iterations for solving the forward kinematic problem of the parallel manipulator. From Figure 7, it is also shown that the proposed DEMPSO has successfully bypassed the steady state of MPSO^2 and inherited its steepest descent properties.

To validate the effectiveness of the proposed DEMPSO algorithm, in our second simulation, we randomly select five poses in the workspace and calculate the relevant leg lengths through inverse kinematics. DEMPSO was employed to search for the optimum pose of the 3-RPS moving platform for the given leg lengths. Table 6 shows the final results for the five different pose situations of the 3-RPS parallel manipulator. From the table, it can be seen that the searched pose value has approached the real pose value with a very small error value ($\approx 1e-5$) when the termination fitness value E_2 is set as $1e-8$. The computation time is almost the same for different pose situations.

5. Conclusions

In this paper, a hybrid strategy combined with DE and MPSO, termed DEMPSO, is developed to get the numerical solution of parallel manipulator forward kinematics. The proposed hybrid method benefits from the efficient global optimization of DE and the fast convergence rate of MPSO; meanwhile, it avoids the possible local optimization of MPSO. Using this method, the search bounds can be narrowed by the DE-based algorithm subtly; afterwards, the MPSO, with its fast rate of convergence, can obtain the global optimization in this narrowed search space.

Table 4: Simulation results of forward kinematics with DEMPSO for different fitness values of DE.

$E1$	$E2$	Iteration of DE	Iteration of MPSO2	Break time of DE (sec)	Total time of DEMPSO (sec)
80	$1e-08$	225	1030	0.017000	0.182367
75	$1e-08$	224	77	0.032000	0.099333
70	$1e-08$	235	81	0.031000	0.096767
60	$1e-08$	246	84	0.047000	0.098933
50	$1e-08$	281	62	0.050000	0.104567
10	$1e-08$	400	68	0.060000	0.104100
1	$1e-08$	688	42	0.060000	0.127433
0.5	$1e-08$	796	42	0.080000	0.142100

Table 5: Comparison results of forward kinematics with DEMPSO, DEPSO, DE, and PSO.

	Pop. number	Fitness error	Iteration	Time (sec)
DEMPSO	30	$1e-08$	267	0.076401
DEPSO	30	$1e-08$	2628	2.244032
DE	30	$1e-08$	3120	0.859791
PSO	30	$1e-08$	1137	1.608181
MPSO[1]	30	$1e-08$	560	1.119108
MPSO[2]	30	$1e-08$	2433	3.561454

Table 6: Simulation results for different situations with DEMPSO.

Number	Length of leg (mm)	Real pose	DEMPSO	Error $* 10^{-4}$	Fitness error	Time (sec)
1	$L1$: 499.018	α^r: 5°	α^s: 4.9999°	0.4142	$7.6010e-08$	0.081441
	$L2$: 557.731	β^r: 12°	β^s: 12.0000°	0.8340		
	$L3$: 534.303	$Z_e{}^r$: 517 mm	$Z_e{}^s$: 517.0000	0.2991		
2	$L1$: 489.714	α^r: −3°	α^s: −3.0000°	0.0303	$3.4300e-08$	0.135116
	$L2$: 506.550	β^r: 6°	β^s: 6.0000°	0.1344		
	$L3$: 520.611	$Z_e{}^r$: 492 mm	$Z_e{}^s$: 491.9999	1.0628		
3	$L1$: 701.251	α^r: 13°	α^s: 13.0000°	−0.1310	$6.2684e-08$	0.087767
	$L2$: 656.309	β^r: −18°	β^s: −18.0000°	0.09136		
	$L3$: 600.050	$Z_e{}^r$: 641 mm	$Z_e{}^s$: 640.9999	−1.4288		
4	$L1$: 613.344	α^r: −12°	α^s: −12.0000°	−0.2467	$5.7662e-08$	0.099704
	$L2$: 558.392	β^r: −7°	β^s: −7.0000°	0.4117		
	$L3$: 613.051	$Z_e{}^r$: 583 mm	$Z_e{}^s$: 583.0001	1.0263		
5	$L1$: 655.862	α^r: 16°	α^s: 16.0000°	0.2104	$8.8969e-08$	0.116486
	$L2$: 767.441	β^r: 19°	β^s: 19.0001°	0.8855		
	$L3$: 694.392	$Z_e{}^r$: 695 mm	$Z_e{}^s$: 694.9999	−0.8913		

To validate the proposed hybrid optimization method, a 3-RPS parallel manipulator was used as an example to numerically solve the nonlinear forward kinematics. In order to get the position and orientation of the moving platform of the manipulator, the algorithms are utilized to solve the constructed fitness function. The comparison study of the proposed DEMPSO algorithm and the DE, PSO, and MPSO algorithms showed that DEMPSO can obtain the best performance for the numerical solution of parallel manipulator forward kinematics. Regarding the general case of nondifferentiable, nonlinear, and multimodal objective functions, further benchmark simulation should be carried out to verify the universality of the proposed hybrid DEMPSO algorithm.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

References

[1] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] D. Karaboğa and S. Ökdem, "A simple and global optimization algorithm for engineering problems: Differential evolution

algorithm," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 12, no. 1, pp. 53–60, 2004.

[3] J. Kennedy, "Particle swarm: social adaptation of knowledge," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '97)*, 1997.

[4] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer US, Boston, MA, USA, 2011.

[5] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, 1997.

[6] M. Clerc, "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, pp. 1951–1957, Washington, DC, USA, July 1999.

[7] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[8] Y. Luo and Q. Liu, "Chaotic finding method of position forward kinematics of 3-DOF parallel robot," in *Proceedings of the 3rd International Symposium on Intelligent Information Technology Application, IITA 2009*, pp. 44–47, Shanghai, China, November 2009.

[9] A. Ramezan Shirazi, M. M. Seyyed Fakhrabadi, and A. Ghanbari, "Analysis and optimization of the 5-RPUR parallel manipulator," *Advanced Robotics*, vol. 28, no. 15, pp. 1021–1031, 2014.

[10] H. Zhang, T. Ren, and M. Pazilai, "Forward position solution of 3-RPS in-parallel manipulator based on particle swarm optimization," in *Proceedings of the 26th Chinese Control and Decision Conference, CCDC 2014*, pp. 4171–4177, Changsha, China, June 2014.

[11] K.-M. Lee and D. K. Shah, "Kinematic Analysis of a Three-Degrees-of-Freedom In-Parallel Actuated Manipulator," *IEEE Journal on Robotics and Automation*, vol. 4, no. 3, pp. 354–360, 1988.

[12] Y. Fang and Z. Huang, "Kinematics of a three-degree-of-freedom in-parallel actuated manipulator mechanism," *Mechanism and Machine Theory*, vol. 32, no. 7, pp. 789–796, 1997.

[13] G. Abbasnejad, S. Zarkandi, and M. Imani, "Forward kinematics analysis of a 3-PRS parallel manipulator," *World Academy of Science, Engineering and Technology*, vol. 37, pp. 329–335, 2010.

[14] J. Gallardo, H. Orozco, and J. M. Rico, "Kinematics of 3-RPS parallel manipulators by means of screw theory," *The International Journal of Advanced Manufacturing Technology*, vol. 36, no. 5-6, pp. 598–605, 2008.

[15] S. S. Shi, Y. T. Song, Y. Cheng et al., "Design and implementation of storage cask system for EAST articulated inspection arm (AIA) robot," *Journal of Fusion Energy*, vol. 34, no. 4, article no. A008, pp. 711–716, 2015.

[16] G. Cui, H. Zhang, D. Zhang, and F. Xu, "Analysis of the kinematic accuracy reliability of a 3-DOF parallel robot manipulator," *International Journal of Advanced Robotic Systems*, vol. 12, 2015.

[17] S. Shi, Y. Song, Q. Yang et al., "Numerical analysis of MD events and preliminary thermal calculation for KTX vacuum vessel," *Fusion Engineering and Design*, vol. 88, no. 12, pp. 3180–3184, 2013.

[18] R. Chandra and L. Rolland, "Global–local population memetic algorithm for solving the forward kinematics of parallel manipulators," *Connection Science*, vol. 27, no. 1, pp. 22–39, 2015.

[19] W. Wang, Q. Li, F. He, and P. Allaire, "Numerical and experimental stability investigation of a flexible rotor on two different tilting pad bearing configurations," *International Journal of Rotating Machinery*, vol. 2014, Article ID 697925, 11 pages, 2014.

[20] X.-S. Wang, M.-L. Hao, and Y.-H. Cheng, "On the use of differential evolution for forward kinematics of parallel manipulators," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 760–769, 2008.

[21] N. N. Son and P. H. A. Ho, "Adaptive MIMO Neural Network Model Optimized by Differential Evolution Algorithm for Manipulator Kinematic System Identification," 2014.

[22] Z.-F. Hao, G.-H. Guo, and H. Huang, "A particle swarm optimization algorithm with differential evolution," in *Proceedings of the International Conference on Machine Learning and Cybernetics*, pp. 1031–1035, IEEE, Hong Kong, China, August 2007.

[23] M. Pant, R. Thangaraj, C. Grosan, and A. Abraham, "Hybrid differential evolution - particle swarm optimization algorithm for solving global optimization problems," in *Proceedings of the 3rd International Conference on Digital Information Management, ICDIM 2008*, pp. 18–24, London, UK, November 2008.