Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 1921479, 10 pages
https://doi.org/10.1155/2017/1921479

Research Article

Solving the Time-Jerk Optimal Trajectory Planning Problem of a Robot Using Augmented Lagrange Constrained Particle Swarm Optimization

Shaotian Lu, Jingdong Zhao, Li Jiang, and Hong Liu

State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin 150006, China

Correspondence should be addressed to Jingdong Zhao; 11b908029@hit.edu.cn

Received 15 February 2017; Accepted 10 May 2017; Published 15 June 2017

Academic Editor: Luciano Mescia

The problem of minimum time-jerk trajectory planning for a robot is discussed in this paper. The optimal objective function is composed of two segments along the trajectory, which are the proportional to the total execution time and the proportional to the integral of the squared jerk (which denotes the derivative of the acceleration). The augmented Lagrange constrained particle swarm optimization (ALCPSO) algorithm, which combines the constrained particle swarm optimization (CPSO) with the augmented Lagrange multiplier (ALM) method, is proposed to optimize the objective function. In this algorithm, falling into a local best value can be avoided because a new particle swarm is generated per initial procedure, and the best value gained from the former generation is saved and delivered to the next generation during the iterative search procedure to enable the best value to be found more easily and more quickly. Finally, the proposed algorithm is tested on a planar 3-degree-of-freedom (DOF) robot; the simulation results show that the algorithm is effective, offering a solution to the time-jerk optimal trajectory planning problem of a robot under nonlinear constraints.

1. Introduction

Trajectory planning is the foundation of robot trajectory control. The aim of trajectory planning is to generate trajectory points and plan a smooth trajectory. Optimal trajectory planning has been a focus of robot research studies in recent years because its performance is important for the efficiency and motion stability of a robot. The major steps in optimal trajectory planning are as follows. First, several points along a given geometric path are given. Then, the inverse kinematic solutions of the given trajectory points are obtained. Next, interpolating functions are used to perform trajectory planning in the joint space. Finally, an optimal method is used to optimize the trajectory in order to make it smooth and continuous. Boundary condition constraints, such as those on the velocity, acceleration, jerk, and time, need to be considered in the optimization procedure.

Reasonable trajectory planning can reduce vibration and ensure the long joint life and high path tracking accuracy of a robot. The minimum-time trajectory planning approach was first proposed in the literature to improve the efficiency of robots; the study of such planning has become an important field in trajectory optimization. The purpose of minimum-time trajectory planning is to plan the shortest execution time trajectory according to the given points while the boundary constraints are satisfied. Examples of minimum-time trajectory planning are provided in [1–5]. In fact, vibration may appear when a robot is in the moving process or when it is stopped, which will negatively impact the tracking accuracy of the robot; therefore, minimum vibration is an important performance index, and many researchers have studied how to utilize minimum-jerk trajectory planning to reduce vibration [6–8]. The minimum-time and minimum-jerk trajectory planning studies are often considered together to improve the efficiency and smoothness of the trajectory and reduce the vibration of a robot; the combination of these two methods is referred to as time-jerk optimal trajectory planning. Examples of such trajectory planning are provided in [9–13].

In the many aforementioned optimal trajectory planning studies, the main research procedure can be summarized as follows. First, the optimal objective function is transformed into a mathematic expression. Second, relational optimization methods are adopted to optimize the mathematic expression. Finally, the optimal results are compared, and a conclusion is reached considering the requirements. Currently, the major interpolated functions used in joint space include cubic splines and fifth-order B-splines, and the major optimal methods include genetic algorithms, PSO, and the sequential quadratic programming (SQP) strategies. The current literature regarding time-jerk optimal trajectory planning suffers from two problems. First, few studies have been performed on this topic, and further in-depth study is required. Second, because optimal trajectory planning is an optimization problem with constraints, the penalty function method is typically used when considering these issues. In the penalty function method, in certain cases, the nonlinear constrained problem can only converge to the optimal solution when the penalty factor becomes infinitely large (according to the exterior penalty function method) or infinitely small (according to the interior penalty function method). Moreover, a large number of tests need to be performed to determine a reasonable initial penalty factor. Therefore, although some optimal algorithms adopted in the prior literature have been improved, it is difficult to achieve a balance in terms of the quality and efficiency of the optimal solution because of the condition restrictions.

This paper proposes a new approach for time-jerk optimal trajectory planning in order to increase the efficiency and reduce the vibration of the robot. The new approach, named augmented Lagrange constrained particle swarm optimization (ALCPSO), combines the constriction factor method from the basic PSO algorithm with the ALM. In contrast to previous research, a new particle swarm is generated for each initial procedure of the PSO; this process can avoid falling into the local best value, and the best value of the prior generation is saved and delivered to the next generation during the iterative search for procedure to allow the best value to be found more easily and quickly. This approach enables the best value to be obtained even for finite penalty factors. The new approach does not require an accurate dynamic model or a continuously differentiable objective function; thus, it has stronger robustness.

The remainder of this paper is organized as follows. The minimum time-jerk trajectory planning problem is described in Section 2. Application of the technique based on the CPSO and ALM methods to solve the nonlinear constrained optimization problem is described in Section 3. Section 4 presents the simulation results of the ALCPSO algorithm. Section 5 presents the conclusions of this study.

2. The Minimum Time-Jerk Trajectory Planning Problem

The optimal trajectory planning premise of this paper is that the geometric path has been given. Thus, the joint space position points corresponding to the discrete operative space (Cartesian space) can be obtained according to inverse kinematics; subsequently, the joint trajectory of the robot can also be obtained via the polynomial or the cubic splines method by connecting the relevant position points. The use of cubic splines in trajectory planning is ordinary because the generated trajectories have continuous acceleration values. In addition, in contrast to higher-order polynomials, the cubic splines can overcome excessive oscillations and overshoot problems between any pair of reference points. In this paper, we implement the joint trajectory planning by utilizing the cubic splines described in [10, 11, 14]. The trajectory planning performance depends largely on the objective function. The objective function [9] is defined as follows:

$$
\begin{aligned}
\min \quad & f(h) \\
& = K_T N \sum_{i=1}^{n-1} h_i + \alpha K_J \sum_{i=1}^{N} \sum_{i-1}^{n-1} \left[\frac{\left(\ddot{Q}_{j,i+1} - \ddot{Q}_{j,i} \right)^2}{h_i} \right] \\
\text{s.t.} \quad & \max \left\{ \left| \dot{Q}_{j,i}(t_i) \right|, \left| \dot{Q}_{j,i}(t_i*) \right|, \left| \dot{Q}_{j,i}(t_{i+1}) \right| \right\} - V_{jm} \\
& \leq 0 \\
& \max \left\{ \left| \ddot{Q}_{j,1}(t_1) \right|, \left| \ddot{Q}_{j,2}(t_2) \right|, \ldots, \left| \ddot{Q}_{j,n}(t_n) \right| \right\} \\
& \quad - A_{jm} \leq 0 \\
& \max \left| \frac{\ddot{Q}_{j,i}(t_{i+1}) - \ddot{Q}_{j,i}(t_i)}{h_i} \right| - J_{jm} \leq 0 \\
& \sum_{i=1}^{n-1} h_i - T_m \leq 0 \\
& \qquad j = 1, \ldots, N \ \forall i = 1, \ldots, n-1.
\end{aligned}
\tag{1}
$$

In (1), $K_T + K_J = 1$, through adjusting the values of K_T and K_J to allow the total execution time and jerk effect to reach the optimum. Because the total execution time and the proportional to the integral of the squared jerk may have a large difference in quantity grade, the elastic coefficient is introduced to balance the effects of them. The meanings of the symbols in (1) are presented in the Symbols.

Therefore, the optimal trajectory planning problem under constraints can be solved by using (1) to search for the optimum solution of time interval h_i.

3. Solving the Nonlinear Constrained Optimization Problem

References [15–17] presented the PSO algorithm, which is a theory that explains the origins of the performance of individuals for bird flocks or fish schools about their collective behavior. The ALM method has numerous advantages, such as its numerical stability and calculated efficiency surpassing the penalty function.

3.1. Constrained Particle Swarm Optimization. In [18, 19], the constriction factor χ of PSO was introduced to ensure the convergence of the PSO as follows:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}. \tag{2}$$

The particle velocity and position of the PSO are updated as follows:

$$\begin{aligned} v_j^i(k+1) &= \chi\left[v_j^i(k) + c_1 r_1\left[p_j^i(k) - x_j^i(k)\right]\right. \\ &\quad \left. + c_2 r_2\left[g_j - x_j^i(k)\right]\right], \\ x_j^i(k+1) &= x_j^i(k) + v_j^i(k+1) \\ &\qquad 1 \le i \le n \ \forall 1 \le j \le N_{d,\max}. \end{aligned} \tag{3}$$

In (2), researchers commonly define $\varphi = c_1 + c_2$ and set $c_1 = c_2 = 2.05$; thus, φ becomes 4.1 and $\chi = 0.729$. In (3), k denotes the number of iterations; c_1 and c_2 are nonnegative values named acceleration factors; r_1 and r_2 are two random values distributed in the range $[0, 1]$; $p_j^i(k)$ describes the individual best position; g_j describes the global best position. Now, the PSO can be referred to as the constrained particle swarm optimization (CPSO) algorithm, which can realize an effective balance between the global search and the local search.

3.2. Augmented Lagrange Multiplier (ALM) Method. The ALM method is similar to the Lagrange multiplier method; however, in contrast to the latter, the former does not require many tests to determine a reasonable initial penalty factor, and the dynamic penalty factor does not tend to infinity; that is, the former surpasses the latter in terms of numerical stability and computational efficiency [20–22]. For general inequality constraints, the optimization model can be described as

$$\begin{aligned} \min \quad & f(X) \\ \text{s.t.} \quad & g_j(X) \le 0 \quad j = 1, \ldots, m. \end{aligned} \tag{4}$$

Its ALM method can be defined as follows:

$$L(X, \lambda, r) = f(X) + \sum_{j=1}^{m}\left[\lambda_j \varphi_j + r_j \varphi_j^2\right], \tag{5}$$

where $X = (x_1, \ldots, x_n)$ denotes the independent variable, $f(X)$ denotes the objective function, $g_j(X) \le 0$ $(j = 1, \ldots, m)$ denotes the inequality constraints, λ_j denotes the Lagrange multiplier, and r_j denotes the penalty factor. φ_j is described as follows:

$$\varphi_j = \max\left(g_j(x), -\frac{\lambda_j}{2r_j}\right) \quad j = 1, \ldots, m. \tag{6}$$

The solution of the original constrained problem (4) can be obtained by solving (5). The Lagrange multiplier iterative formula is expressed as

$$\lambda_j^{v+1} = \lambda_j^v + 2r_j\varphi_j, \tag{7}$$

where v denotes the number of times that the Lagrange multiplier is updated. The penalty factor updated formula is expressed as follows:

$$r_i^{v+1} = \begin{cases} 2r_j^v & \left|g_j(X^v)\right| > \left|g_j\left(X^{v-1}\right)\right| \wedge \left|g_j(X^v)\right| > \varepsilon_g \\ \dfrac{r_j^v}{2} & \left|g_j(X^v)\right| \le \varepsilon_g \\ r_j^v & \text{else,} \end{cases} \tag{8}$$

where ε_g denotes the constraint error accuracy.

When $v = 1$, the termination criterions are as follows:

$$\begin{aligned} c = \max\{&\max\left[0, g_j(X^v)\right] \le \varepsilon; \ j = 1, \ldots, m \ \forall i \\ &= 1, \ldots, v_{\max}\}. \end{aligned} \tag{9}$$

When $v = 2, \ldots, v_{\max}$, the termination criterions are defined as follows:

$$\left|f(X^v) - f\left(X^{v-1}\right)\right| \wedge c \le \varepsilon, \tag{10}$$

where ε denotes the convergence accuracy and $v_{\max}$ denotes the update maximal times.

The ALM method can guarantee that the optimal solution can be found when the penalty factor does not reach infinity, which cannot be guaranteed by either the interior or the exterior penalty function method.

3.3. Augmented Lagrange Constrained Particle Swarm Optimization (ALCPSO) Algorithm. Because the CPSO and ALM enjoy their respective advantages, in this paper, we propose an algorithm named ALCPSO algorithm which combines the CPSO with the ALM in order to make use of their respective advantages to provide a new approach for solving nonlinear constrained optimization problems. Time-jerk optimal trajectory planning is such a problem. The ALCPSO algorithm comprises the following steps.

[ALCPSO1]. Set $v_{\max}$, $k_{\max}$, $N_{d,\max}$, ε, ε_g, λ_j^0, r_j^0, n, $k = 0$, and $v = 1$. Transform the constrained problem (4) into the unconstrained problem (5).

[ALCPSO2]. Initialize $N_{d,\max}$ particles with randomly chosen positions and velocities and evaluate the corresponding fitness values at each position.

[ALCPSO3]. Check the termination criterions (9). If they are satisfied, the algorithm terminates with the best value $x^* = x_{\text{swarm}}^{\text{best}}$; if they are not satisfied, the obtained best value

x^* is saved. Initialize $N_{d,\max}$ particles with randomly chosen positions and velocities and set $k = 1$.

[ALCPSO4]. Utilize the obtained best value of former generation to replace one group of particles.

[ALCPSO5]. Apply (3) to update the particles, evaluate the corresponding fitness values at each position, and find the best $p^k_{g\text{-best}}$.

[ALCPSO6]. Check the updated number of times of the Lagrange multiplier/penalty factor. If it is satisfied, check the termination criterions (10); if the termination criterions (10) are satisfied, the best value $x^* = x^{\text{best}}_{\text{swarm}}$ is obtained; if they are not satisfied, update the Lagrange multipliers and the penalty factors according to (7) and (8) and set $v = v+1$. If the updated number of times of the Lagrange multiplier/penalty factor is not satisfied, proceed with step [ALCPSO5].

In this algorithm, a new particle swarm is freshly generated in each initial procedure to avoid falling into the local best value, and the best value can be found more easily and quickly because the best value of the prior generation is saved and delivered to the next generation during the iterative search for procedure.

4. Trajectory Planning Simulation and Discussion

Before optimizing the robot's trajectory, its inverse kinematics should be obtained. In [23, 24], the configuration control approach is proposed to solve the inverse kinematics of the redundant robot; this approach ensures that the inverse kinematics is unique and enables the robot to perform cyclic motion, which is an essential requirement for repetitive operations. Moreover, this approach requires little computational time, making it particularly favorable for the real-time control of a redundant robot. In this paper, we use the damped least squares (DLS) of the configuration control approach to determine the inverse kinematics of a redundant robot. Its formulation can be written in the following form:

$$\dot{q} = W_v^{-1} J^T \left[J W_v^{-1} J^T + \lambda^2 W^{-1} \right]^{-1} \left(\dot{X}_d + Ke \right), \quad (11)$$

where

W is a symmetric positive-definite weighting matrix, $W = \text{diag}[W_e, W_c]$, where W_e and W_c are symmetric positive-definite weighting matrices in accordance with the basic task and an additional task, respectively;

W_v is a symmetric positive-definite weighting matrix;

λ is a positive scalar constant;

K is a symmetric positive-definite feedback gain (constant);

e is error, $e = X_d - X$.

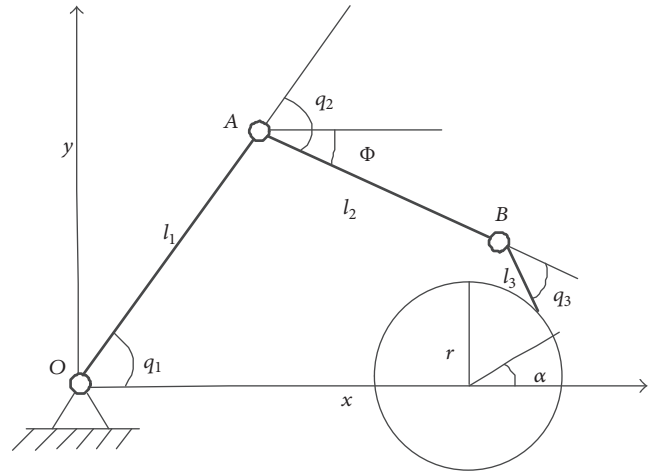

Figure 1: Schematic diagram of the planar 3-DOF robot.

Next, the position solution of inverse kinematics can be obtained by integrating (11). We use the proposed ALCPSO method to perform a trajectory planning simulation of a robot to verify the accuracy and reliability of the method. The program is compiled in Matlab™ R2014. The test object is a space robot under the condition that the position and pose of the associated carrying platform are under control; it is a 3-DOF planar robot shown in Figure 2. The tested mission is tracking a circular path in plane. In Figure 1, q_1, q_2, and q_3 are joint angles. The robot geometric parameters are defined as l_1 = 2,080 mm, l_2 = 2,080 mm, and l_3 = 430 mm. Because the robot has a redundant degree of freedom, the configuration control approach is utilized to determine its inverse kinematics. The initial position of the end effector is (4000, 0) (mm), with corresponding joint angles of (30°, −49.03°, −38.27°), and the radius r and the center of the track circle are 400 mm and (3600, 0) (mm), respectively. The following procedure is used to obtain the inverse kinematics of the robot:

(1) Set the planned trajectory points in the circular path as a function of time t, mean $\alpha(t)$, which is used to plan the desired trajectory points.

(2) Choose $\Phi = q_1 + q_2$ as the additional task.

(3) Let $e = [x_d - x; y_d - y; q_{1d} - q_1]$.

(4) Put the corresponding values into (11) and perform integration to obtain the position solution of inverse kinematics.

In this paper, $\alpha(t)$ indicates the circular central angle; it is divided into uniform acceleration, uniform velocity, and uniform deceleration for a total of three sections along the counterclockwise rotation. The robot starts from an initial position with the angular acceleration from the value of $120/(1/80 \times n)^{2\circ}/s^2$ (n denotes the number of planned via-points). When the robot along the circle moves by 60°, the uniform acceleration section is completed, and it enters the uniform velocity section, moving along the circle by 240°; finally, the robot with value of $120/(1/80 \times n)^{2\circ}/s^2$ enters the

Table 1: Input data for trajectory planning.

Joint	Plan-points (deg)								
	1	2	3	4	5	6	7	8	9
1	30.00	Virtual	39.94	49.24	47.66	37.60	29.48	Virtual	30.04
2	−49.03		−57.25	−72.45	−79.89	−72.42	−57.16		−49.14
3	−38.27		−42.00	−49.16	−52.48	−49.25	−42.30		−37.96

Table 2: Results of trajectory optimization when utilizing the ALCPSO algorithm.

Parameters	Initial	Numerical results ($\alpha = 100$)				
		$K_T = 1$, $K_J = 0$	$K_T = 0.8$, $K_J = 0.2$	$K_T = 0.5$, $K_J = 0.5$	$K_T = 0.2$, $K_J = 0.8$	$K_T = 0$, $K_J = 1$
h_1	7.5	0.2527	2.8834	3.5899	5.0726	4.9810
h_2	7.5	4.8729	5.4128	6.7907	8.1513	8.9584
h_3	7.5	5.2656	6.1875	6.5949	8.3171	8.7707
h_4	7.5	3.2660	5.2699	5.8325	7.4311	7.6433
h_5	7.5	5.1017	4.8675	6.4061	7.9223	8.4227
h_6	7.5	5.2044	5.5472	6.6090	8.2754	8.7030
h_7	7.5	3.9063	4.2672	7.2471	8.9705	9.5342
h_8	7.5	0.2826	2.7232	2.4045	3.0351	2.9867
$\sum h_i$	60	28.1523	37.1587	45.4748	57.1754	60.0000
$\sum_{j=1}^{N}\sum_{i=1}^{n-1}\left(\frac{\left(\alpha_{j,i+1}-\alpha_{j,i}\right)^2}{h_i}\right)$	0.1119	22.6147	0.8664	0.2714	0.0861	0.0678
min $f(h)$	/	84.4568	106.508	81.7840	41.1919	6.783
Time (sec)	1.749	82.125	33.130	30.369	29.0513	40.212

uniform deceleration section and returns to the initial point, thus completing the entire motion. In this paper, set $n = 8$. Other values are defined as follows:

$n = 8$, $W_e = \text{diag}[1, 1]$, $W_c = 1$, $W_v = \text{diag}[1, 1, 1]$, $K = \text{diag}[1, 1, 1]$, $\lambda = 1$, and $\dot{X}_d = \text{diag}[1, 1, 1]$.

The values of the plan-points of the trajectory are shown in Table 1. As shown in Table 1, the 1-point coincides with the 9-point when the end effector completes a cyclic motion. Thus, there are 9 plan-points and 8 via-points. The kinematic constraints of the velocity, acceleration, and jerk of the joints are 3 deg/s, 3 deg/s^2, and 5 deg/s^3, respectively.

We set the robot's execution time constraint $t = 60$ s, the initial Langrage multiplier $\lambda^0 = (0, 0, \ldots, 0)$, the initial penalty factor $r^0 = (1000, 1000, \ldots, 1000)$, the convergence accuracy $\varepsilon = 0.0001$, the constraint error accuracy $\varepsilon_g = 0.0001$, the Lagrange multiplier update maximal times $v_{\max} = 100$, the maximum number of iterations in the CPSO $k_{\max} = 100$, the swarm particle number $N_{d,\max} = 40$, and the elastic coefficient $\alpha = 100$. The weighting coefficients are adjusted as follows: K_T is 1, 0.8, 0.5, 0.2, and 0, and the corresponding K_J can be determined via calculation. We performed the proposed optimal method using the five weighting coefficient values, and the corresponding results are shown in Table 2. The execution time gradually increases when K_T gradually decreases from 1 to 0. The joint trajectories prior to optimization are shown in Figure 2.

The simulation results when K_T is 1, 0.5, and 0 are shown in Figures 3–5, respectively. When K_T is 1, the objective function contains only the minimum-time function. The joints' velocities, accelerations, and jerks shown in Figure 3 are generally larger than those in Figure 2; the joint maximum jerk reaches the upper bounds on the absolute value at the initial moment, which corresponds to a rough trajectory, resulting in a comparatively larger vibration and lower path tracking accuracy of the robot during the moving process. If the robot operates in such a condition for an excessive period of time, its lifespan will be reduced. However, the working efficiency of the robot can increase because the execution time is reduced; this condition suits a robot with strict working efficiency requirement. The execution time in Figure 4 is longer than that in Figure 3; however, the jerks on the whole are smaller than that in Figure 3. In Figure 5, the execution time is equal to the set constraint t, and compared to the results in Figure 2, the joints' velocities, accelerations, and jerks are generally smaller than those in Figure 2, indicating that, under the same execution time condition, it is beneficial for a robot during the moving process compared with that before performing optimization because of the lower velocities, accelerations, and jerks.

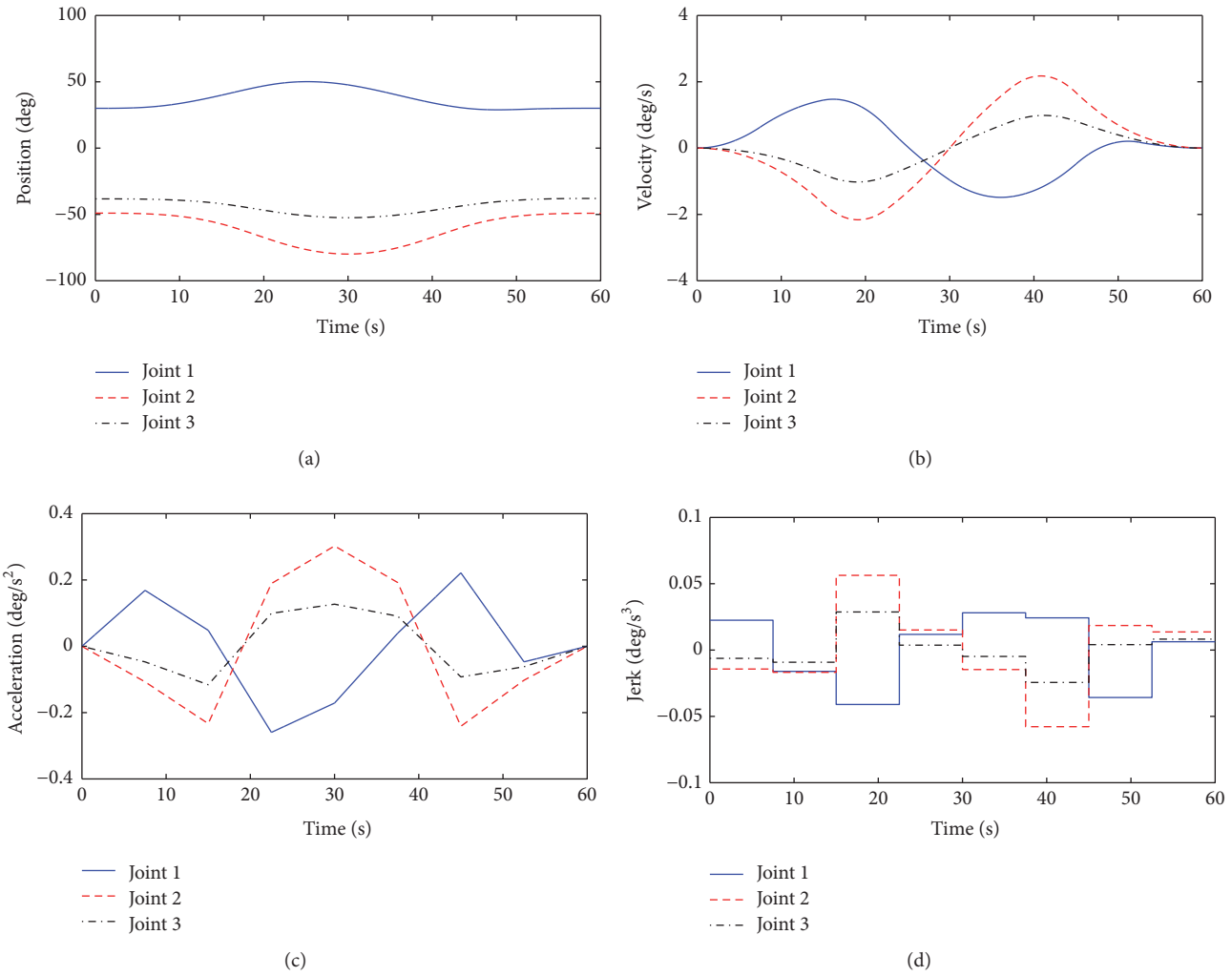

FIGURE 2: The joint trajectories prior to optimization.

From analyzing Table 2 and Figures 2–5, we can achieve relevant objects by adjusting elastic and weighting coefficients according to different requirements.

When we combine the CPSO with the interior penalty function method, the constrained problem (4) can be expressed as follows:

$$L(X, r) = f(X) - r\sum_{j=1}^{m} \ln\left[-g_j(x)\right], \tag{12}$$

where r denotes the penalty factor. We set the initial penalty factor $r^0 = 100000$ and the convergence accuracy $\varepsilon = 0.001$; other corresponding values are referred to in the aforementioned ALCPSO method.

The termination criterions are defined as follows:

$$\left|l\left(x^*\left(r^{(v)}\right), r^{(v)}\right) - f\left(x^*\left(r^{(v)}\right)\right)\right| \le \varepsilon \quad v = 1, \ldots, v_{\max}, \tag{13}$$

where v denotes the number of iterations. The simulation results are shown in Table 3 for typical values of K_T and K_J. After comparing the corresponding results with those of the ALCPSO method, we can find that the ALCPSO method indeed has some advantages, such as the calculated velocity and some obtained best values. In Table 3, when K_T is set as 0, the obtained time is shorter than that of the ALCPSO method. This implies that when K_T is set as 0, the CPSO and interior penalty function methods can be used to perform time-jerk optimal trajectory planning because they can get better time.

When we utilize the SQP method (e.g., the fmincon function of Matlab) described in [9, 10], the initial input is the same as the aforementioned ALCPSO method to perform time-jerk optimal trajectory planning. The simulation results are shown in Table 4. After comparing the corresponding results with those of the ALCPSO method, we can find that the ALCPSO method generally surpasses the SQP method in the obtained best values.

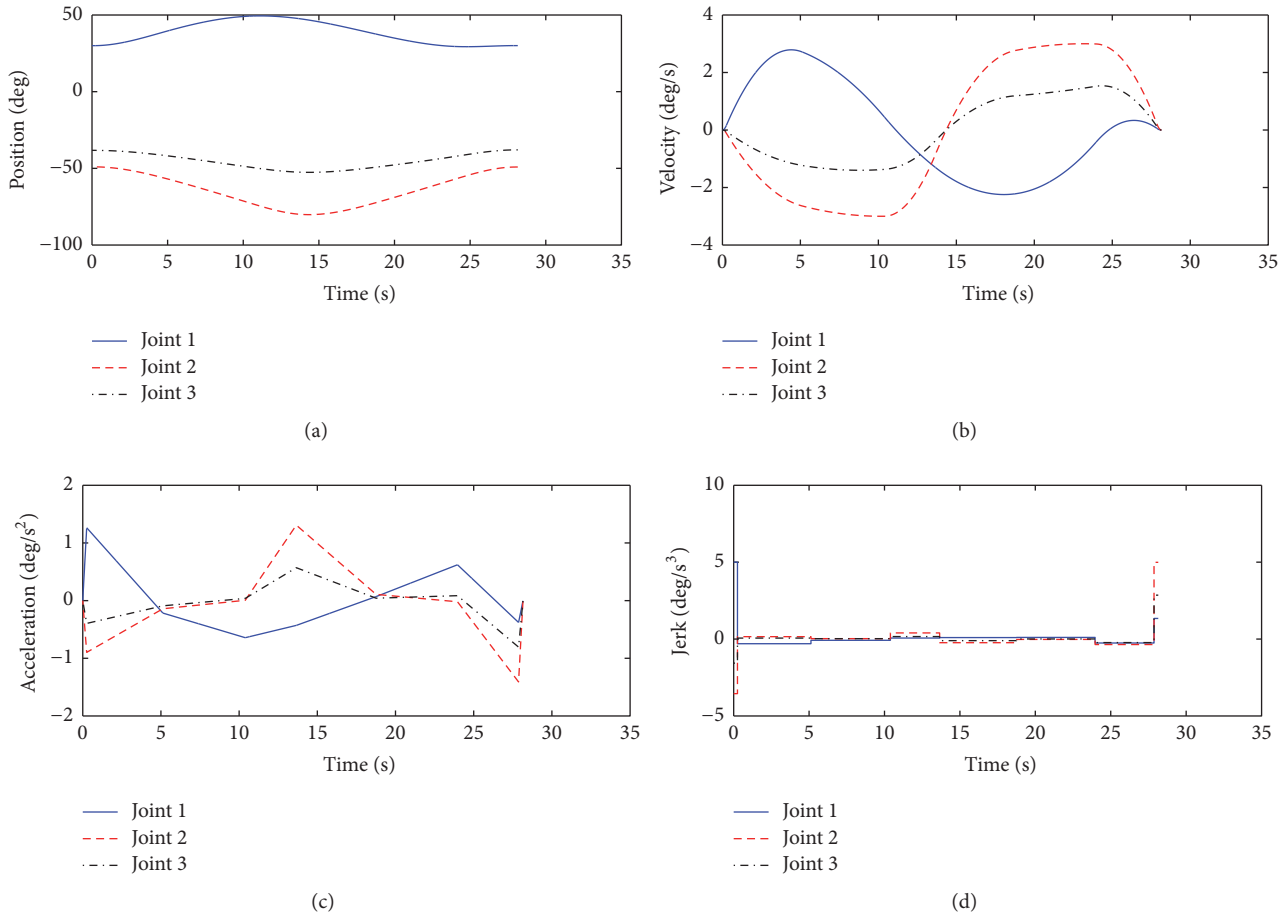

FIGURE 3: Time-jerk optimal joint trajectories when $K_T = 1$ and $K_J = 0$.

TABLE 3: Results of the trajectory optimization when combining the CPSO with the interior penalty function method.

Parameters	Numerical results ($\alpha = 100$)	
	$K_T = 1$	$K_T = 0$
	$K_J = 0$	$K_J = 1$
h_1	4.8291	6.6197
h_2	6.1574	7.0985
h_3	8.0744	7.9880
h_4	4.9833	6.0594
h_5	7.7239	8.0459
h_6	7.8541	7.9070
h_7	4.6968	5.6325
h_8	4.2933	4.2614
$\sum h_i$	48.6122	53.6122
$\sum_{j=1}^{N}\sum_{i=1}^{n-1}\left(\frac{(\alpha_{j,i+1}-\alpha_{j,i})^2}{h_i}\right)$	0.2986	0.1422
min $f(h)$	145.8367	14.2223
Time (sec)	10534.807	10648.469

5. Conclusion

This paper proposed an ALCPSO algorithm, which is based on the CPSO and the ALM methods, to realize the time-jerk optimal trajectory planning of a robot. In this algorithm, a new particle swarm is generated each time in the initial process in order to avoid falling into a local best value. In this manner, the best value can be found more easily and quickly because the best value of the previous generation is saved and delivered to the next generation during the iterative search for procedure. The optimal objective function considers both the total execution time and the integral of the squared jerk along the entire trajectory and then varies the values of two weighting coefficients and the elastic coefficient to achieve such requirements as rapid execution, a smooth trajectory, or both. The algorithm is implemented on a planar 3-DOF robot. The simulation results illustrate that the time-jerk optimal trajectory that meets both the kinematics and the execution time constraints can be obtained. The proposed algorithm is developed without performing an experiment; thus, future work will be devoted to performing experiments to verify the present algorithm's effectiveness.

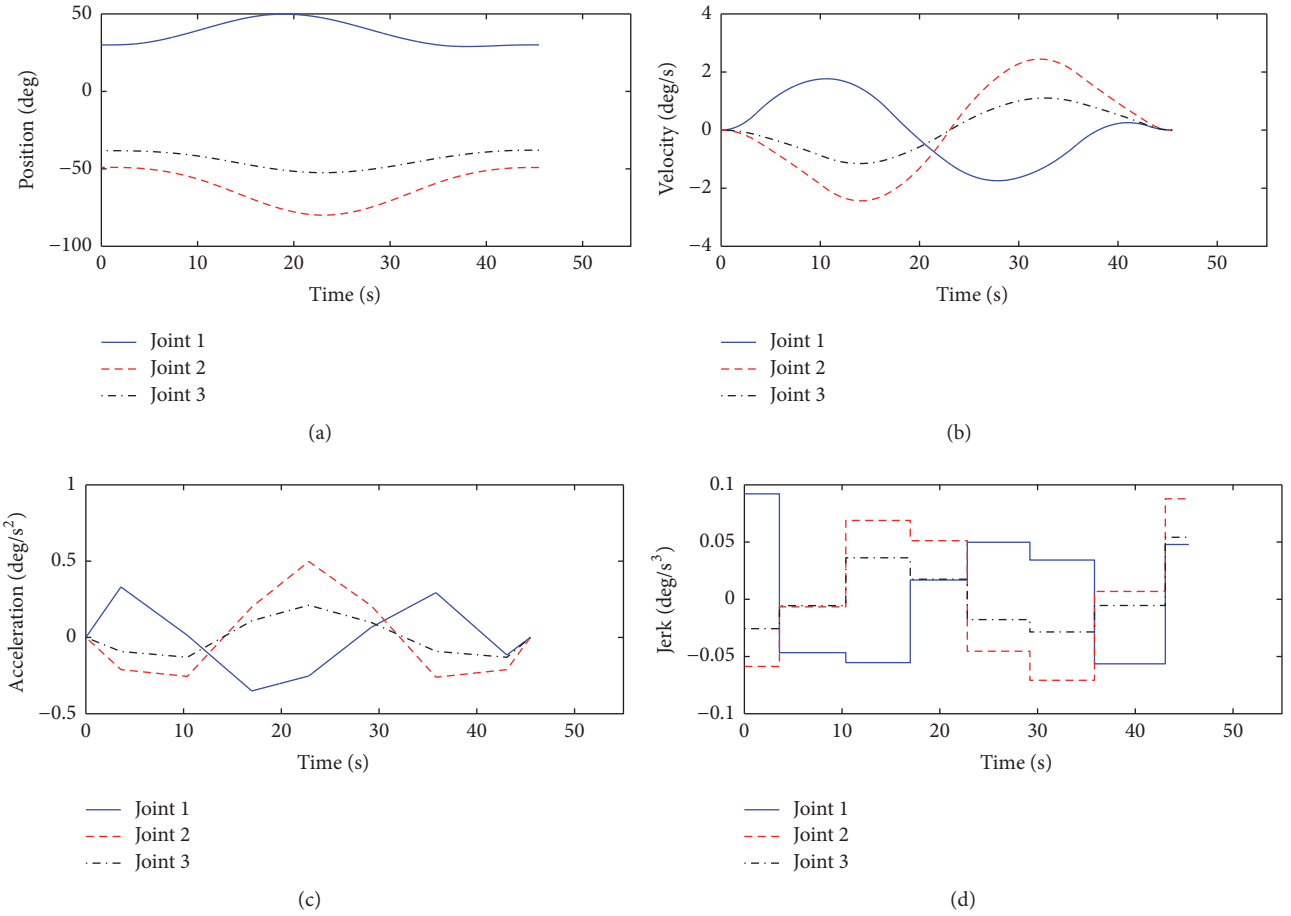

FIGURE 4: Time-jerk optimal joint trajectories when $K_T = 0.5$ and $K_J = 0.5$.

TABLE 4: Simulation results when utilizing the SQP algorithm.

Parameters	Initial	Numerical results ($\alpha = 100$)				
		$K_T = 1$, $K_J = 0$	$K_T = 0.8$, $K_J = 0.2$	$K_T = 0.5$, $K_J = 0.5$	$K_T = 0.2$, $K_J = 0.8$	$K_T = 0$, $K_J = 1$
$\sum h_i$	60	28.4283	37.3813	45.6751	57.2908	59.9794
$\sum_{j=1}^{N}\sum_{i=1}^{n-1}\left(\frac{\left(\alpha_{j,i+1}-\alpha_{j,i}\right)^2}{h_i}\right)$	0.1119	16.6042	0.8090	0.2678	0.0873	0.0689
min $f(h)$	/	85.2850	105.896	81.9050	41.3599	6.8931

Symbols

N: Number of robot joints
K_T: Time weighting coefficient
K_J: Jerk weighting coefficient
h_i: Time interval between two via-points
J_{jm}: Jerk limit for the jth joint (symmetrical)
n: Number of via-points
$\dot{Q}_j(t)$: Velocity of the jth joint
$\ddot{Q}_j(t)$: Acceleration of the jth joint
$\dddot{Q}_j(t)$: Jerk of the jth joint
V_{jm}: Velocity limit for the jth joint (symmetrical)
A_{jm}: Acceleration limit for the jth joint (symmetrical)
T_m: Execution time limit.

Conflicts of Interest

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

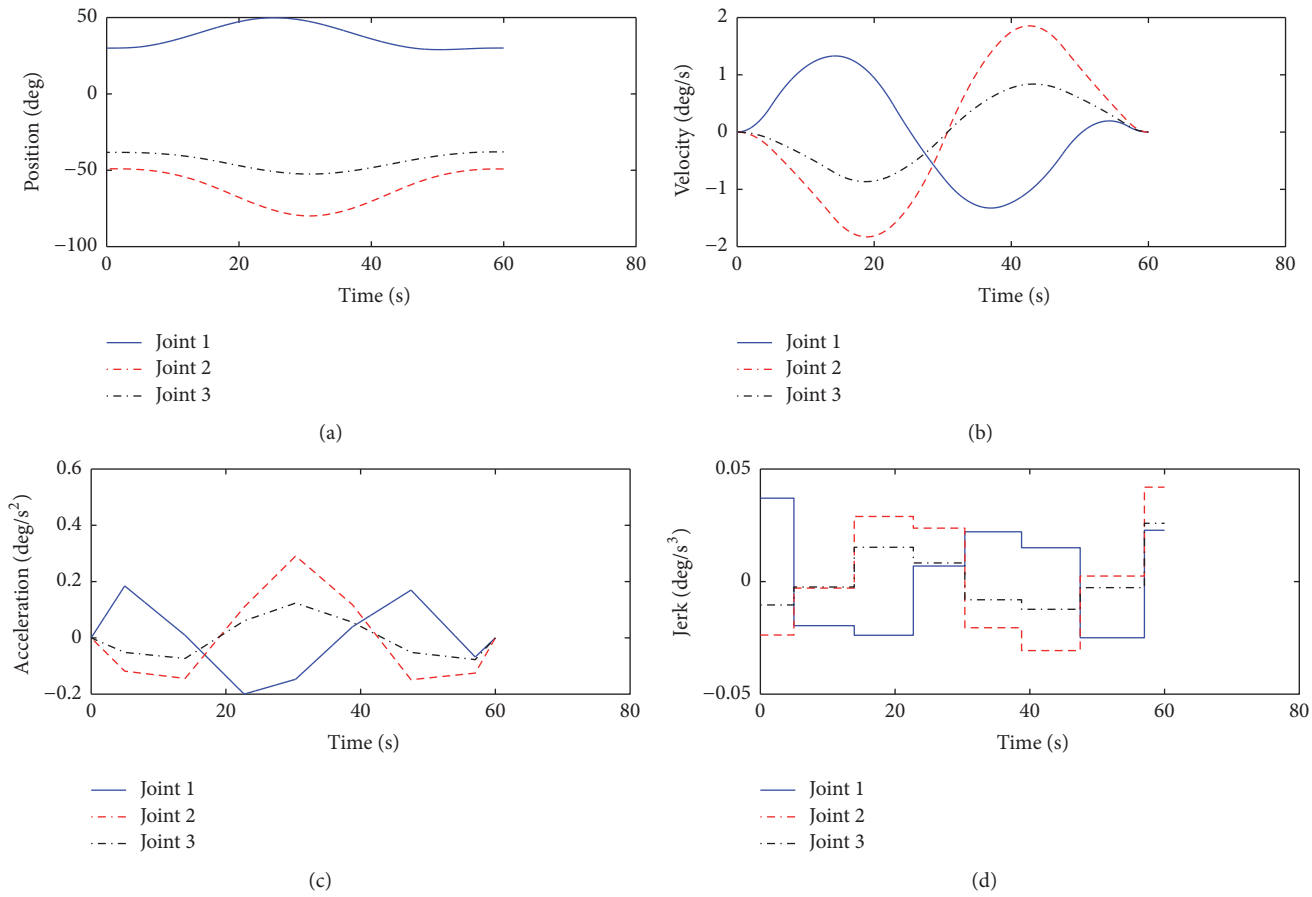

FIGURE 5: Time-jerk optimal joint trajectories when $K_T = 0$ and $K_J = 1$.

Acknowledgments

This research was supported by the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant no. 51521003).

References

[1] C.-S. Lin, P.-R. Chang, and J. Y. S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, no. 12, pp. 1066–1074, 1983.

[2] D. G. Li, J. Y. Qiu, and B. You, "Optimal algorithm for trajectory planning of the robot," *Electric Machines and Control*, vol. 13, no. 1, pp. 123–127, 2009.

[3] A. Piazzi and A. Visioli, "Global minimum-time trajectory planning of mechanical manipulators using interval analysis," *International Journal of Control*, vol. 71, no. 4, pp. 631–652, 1998.

[4] J. Kim, S.-R. Kim, S.-J. Kim, and D.-H. Kim, "A practical approach for minimum-time trajectory planning for industrial robots," *Industrial Robot*, vol. 37, no. 1, pp. 51–61, 2010.

[5] F. Rubio, F. Valero, J. Sunyer, and J. Cuadrado, "Optimal time trajectories for industrial robots with torque, power, jerk and energy consumed constraints," *Industrial Robot*, vol. 39, no. 1, pp. 92–100, 2012.

[6] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 47, no. 1, pp. 140–149, 2000.

[7] A. Piazzi and A. Visioli, "Interval algorithm for minimum-jerk trajectory planning of robot manipulators," in *Proceedings of the 1997 36th IEEE Conference on Decision and Control*, pp. 1924–1927, December 1997.

[8] H.-I. Lin, "A fast and unified method to find a minimum-jerk robot joint trajectory using particle swarm optimization," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 75, no. 3-4, pp. 379–392, 2014.

[9] Z. Cao, H. Wang, W. Wu, and H. Xie, "Time-jerk optimal trajectory planning of shotcrete manipulators," *Journal of Central South University (Science and Technology)*, vol. 44, no. 1, pp. 114–121, 2013.

[10] A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajectories," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 3, pp. 415–426, 2008.

[11] V. Zanotto, A. Gasparetto, A. Lanzutti, P. Boscariol, and R. Vidoni, "Experimental validation of minimum time-jerk algorithms for industrial robots," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 64, no. 2, pp. 197–219, 2011.

[12] M. Cong, X. Xu, and P. Xu, "Time-jerk synthetic optimal trajectory planning of robot based on fuzzy genetic algorithm," in *Proceedings of the 15th International Conference on Mechatronics and Machine Vision in Practice (M2VIP'08)*, pp. 274–279, Auckland, New Zealand, December 2008.

[13] L. Feifei and L. Fei, "Time-jerk optimal planning Of industrial robot trajectories," *International Journal of Robotics and Automation*, vol. 31, no. 1, pp. 1–7, 2016.

[14] S. F. Saramago and J. Steffen, "Optimization of the trajectory planning of robot manipulators taking into account the dynamics of the system," *Mechanism and Machine Theory*, vol. 33, no. 7, pp. 883–894, 1998.

[15] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[16] W. Xu, C. Li, B. Liang, Y. Liu, and Y. Xu, "The Cartesian path planning of free-floating space robot using particle swarm optimization," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 301–310, 2008.

[17] P. Huang and Y. Xu, "PSO-based time-optimal trajectory planning for space robot with dynamic constraints," in *Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics (ROBIO'06)*, pp. 1402–1407, Kunming, China, December 2006.

[18] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in Solar Photovoltaic system," *Journal Applied Soft Computing*, vol. 13, no. 5, pp. 2997–3006, 2013.

[19] M. Wang, P. Wang, J.-S. Lin, X. Li, and X. Qin, "Nonlinear inertia classification model and application," *Mathematical Problems in Engineering*, vol. 2014, Article ID 987686, 9 pages, 2014.

[20] Y. Yu, X. Yu, and Y. Li, "Solving engineering optimization problem by Augmented Lagrange particle swarm optimization," *Journal of Mechanical Engineering*, vol. 45, no. 12, pp. 167–172, 2009.

[21] X. Yan, Y. Zhong, G. Sun, and Z. Zhong, "Cost optimization design of hybrid composite flywheel rotor," *Journal of Mechanical Engineering*, vol. 48, no. 12, pp. 118–126, 2012.

[22] K. Sedlaczek and P. Eberhard, "Using augmented Lagrangian particle swarm optimization for constrained problems in engineering," *Structural and Multidisciplinary Optimization*, vol. 32, no. 4, pp. 277–286, 2006.

[23] H. Seraji, "Configuration control of redundant manipulators: theory and implementation," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, pp. 472–490, 1989.

[24] K. Glass, R. Colbaugh, D. Lim, and H. Seraji, "On-line collision avoidance for redundant manipulators," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 36–43, Atlanta, Ga, USA, 1993.