Robotics and Computer-Integrated Manufacturing 26 (2010) 180–189

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcim

Design optimization of a spatial six degree-of-freedom parallel manipulator based on artificial intelligence approaches

Zhen Gao [a], Dan Zhang [a,*], Yunjian Ge [b]

[a] Faculty of Engineering and Applied Science, University of Ontario Institute of Technology, Oshawa, Ontario L1H 7K4, Canada
[b] Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei, Anhui Province, 230031, P.R. China

ARTICLE INFO

Article history:
Received 26 March 2008
Received in revised form
7 April 2009
Accepted 1 July 2009

Keywords:
Optimization design
System stiffness
Dexterity
Genetic algorithms
Artificial neural networks

ABSTRACT

Optimizing the system stiffness and dexterity of parallel manipulators by adjusting the geometrical parameters can be a difficult and time-consuming endeavor, especially when the variables are diverse and the objective functions are excessively complex. However, optimization techniques that are based on artificial intelligence approaches can be an effective solution for addressing this issue. Accordingly, this paper describes the implementation of genetic algorithms and artificial neural networks as an intelligent optimization tool for the dimensional synthesis of the spatial six degree-of-freedom (DOF) parallel manipulator. The objective functions of system stiffness and dexterity are derived according to kinematic analysis of the parallel mechanism. In particular, the neural network-based standard backpropagation learning algorithm and the Levenberg–Marquardt algorithm are utilized to approximate the analytical solutions of system stiffness and dexterity. Subsequently, genetic algorithms are derived from the objective functions described by the trained neural networks, which model various performance solutions. The multi-objective optimization (MOO) of performance indices is established by searching the Pareto-optimal frontier sets in the solution space. Consequently, the effectiveness of this method is validated by simulation.

1. Introduction

Compared with conventional serial manipulators, parallel manipulators have significant advantages, such as improved stiffness, increased payload, and high force/torque capacity. Moreover, they also have simpler inverse kinematics, which is an advantage for real-time control. Recently, parallel manipulators have been developed for applications in aircraft simulators [1,2], telescopes [3], positioning trackers [4], micro-motion [5–7], and machine tools [8–14]. However, because the theories and technologies for parallel manipulators are still in the early stages of development, most existing parallel manipulators are expensive devices that provide less accuracy than conventional machines. Therefore, further investigation is necessary for making parallel manipulators more attractive to industries [15].

Since the capabilities of current parallel manipulators are limited in more extensive applications, such as biotechnology and automotive manufacturing, performance improvement is one of the most important issues that need to be addressed. The purpose of optimization design aims at enhancing the performance indexes by adjusting the structural parameters, such as the link length, the radii of fixed and moving platforms, and the distance between platform centers. This approach is called the dimensional synthesis-based performance optimization of parallel manipulators. In the optimum design process, several performance indices are involved, such as stiffness, dexterity, accuracy, and workspace.

Many scholars have studied the optimum design of robot manipulators [16–19]. Zhao et al. [20] exploited the least number method for variables in order to optimize the leg length of a spatial parallel manipulator for the purpose of obtaining a dexterous workspace. Furthermore, Stock and Miller [21] presented a method for the multi-dimensional kinematic optimization of the geometry for the linear delta robot architecture. Specifically, they formulated a utility objective function, incorporating two performance indices: manipulability and space utilization. Kucuk and Bingul [22] optimized the workspace of two spherical three-link robot manipulators using the local and global performance indices. Lastly, Ceccarelli and Lanni [23] investigated the multi-objective optimization (MOO) problem of a general 3R manipulator for prescribed workspace limits by utilizing an algebraic formulation. Overall, it is apparent that artificial intelligence technologies provide effective approaches for investigating this topic.

As the primary components of artificial intelligence approaches, genetic algorithms and artificial neural networks play

* Corresponding author at: Faculty of Engineering and Applied Science, University of Ontario Institute of Technology Oshawa, Ontario, L1H 7K4, Canada. Tel.: +1905 721 8668x2965; fax: +1905 721 3370.
E-mail address: Dan.Zhang@uoit.ca (D. Zhang).

0736-5845/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2009.07.002

important roles in various fields of science and technology. In this research, the two components are simultaneously utilized as the optimization criteria for the dimensional synthesis of a symmetrical six-degree-of-freedom (DOF) parallel manipulator. The two main performance indices, system stiffness and dexterity, will be optimized as both single-objective optimization (SOO) and MOO issues to demonstrate the validity of the proposed integrated artificial intelligence approaches and to improve the manipulation capabilities of parallel manipulators.

In this paper, many parameters and complex matrix computations need to be managed. Hence, it is difficult to search for the objective values, or optimal configuration, and the corresponding structural variables, based on the analytical expressions of system stiffness and dexterity. Moreover, with traditional optimization methods, only a few geometric variables could be managed due to the lack of convergence in more complex problems. Genetic algorithms, as powerful and broadly applicable search methods, follow the Darwinian evolutionary principle of "survival-of-the-fittest," where strong traits are retained in the population and weak traits are eliminated. Thus, genetic algorithms can avoid the problems associated with local optima [24], and are therefore suitable for addressing the convergence problem in this scenario. On the other hand, neural networks possess the capability of complex function approximation and generalization by simulating the basic functionality of the human nervous system in an attempt to capture some of its computational strengths. Since the objective function must be solved before using genetic algorithms, neural networks will be utilized to represent expressions of the solutions for the two performance indices of a six-DOF parallel manipulator. For the MOO problem, an algorithm that exhibits compromise should be executed, since it is impossible to maximize or minimize all of the objective function values if they conflict with one another. This methodology will not only provide effective guidance but will also present a new approach for the dimensional synthesis of optimal design in general parallel mechanisms.

The remainder of the paper is organized into distinct sections. Section 2 presents geometric modeling and kinematic analysis, where solutions for the inverse kinematics model and the Jacobian matrix are derived. In Section 3, the optimization criteria of system stiffness and dexterity are illustrated, and the models of performance indexes are deduced. Subsequently, Section 4 discusses the results from the applications of genetic algorithms and neural networks for optimizing the performance indices of the parallel manipulator. Specifically, both the SOO and MOO issues are addressed. Finally, Section 5 presents the conclusion and suggests future work.

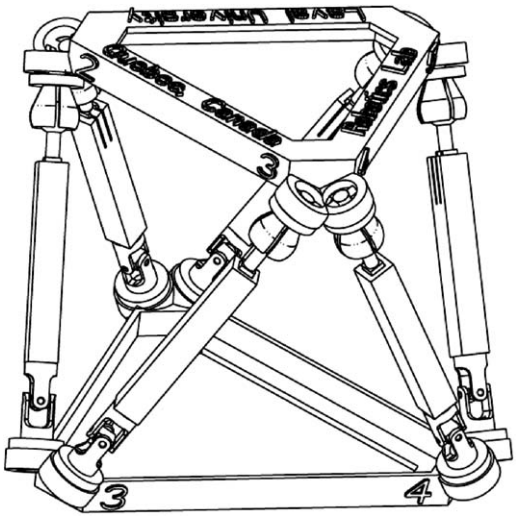

Fig. 1. CAD model of the spatial six-DOF parallel manipulator (by Thierry Laliberte and Gabriel Cote).

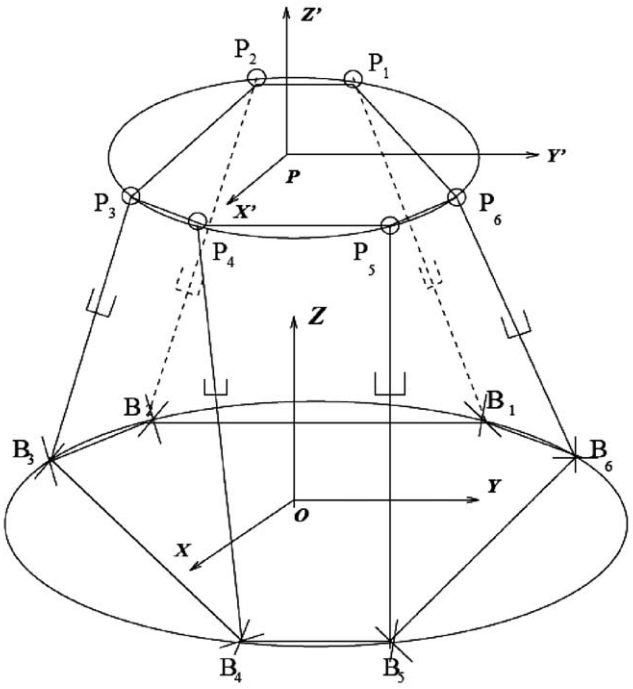

Fig. 2. Schematic representation of the spatial six-DOF parallel mechanism.

2. Geometric modeling and kinematic analysis

2.1. Geometric modeling

In this work, a six-DOF parallel mechanism and its joint distributions on both the base and the platform are shown in Figs. 1–3. This mechanism consists of six identical extensible links connecting the fixed base to a moving platform. The kinematic chains associated with the six legs, from base to platform, consist of a fixed Hooke joint, a moving link, an actuated prismatic joint, a second moving link, and a spherical joint attached to the moving platform. It is also assumed that the vertices on the base and platform are located in the circles of radii R_b and R_p, respectively.

A fixed reference frame, O–xyz, is connected to the base of the mechanism, and a moving coordinate frame, P–$x'y'z'$, is attached to the platform. In Fig. 2, the attachment points of the actuated legs to the base are represented by B_i and the attachment points of

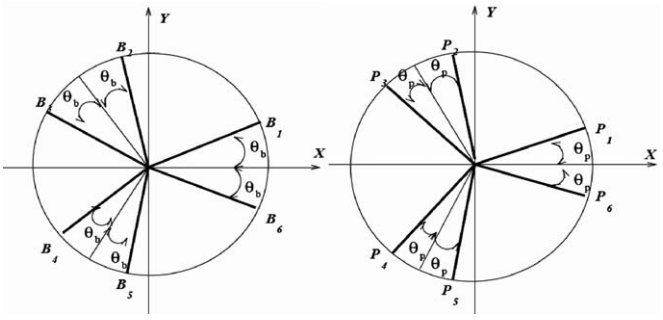

Fig. 3. Position of the attachment points: on the base (left), and on the platform (right).

all legs to the platform are represented by P_i, with $i = 1, \ldots, 6$, while point P is located at the center of the platform with the coordinate of $P(x,y,z)$.

The Cartesian coordinates of the platform are given by the position of point P with respect to the fixed frame, and the orientation of the platform, or the orientation of the frame P–$x'y'z'$ with respect to the fixed frame, is represented by three Euler angles ϕ, θ, and ψ, or by the rotation matrix $\boldsymbol{Q}$.

2.2. Inverse kinematics

The inverse kinematics problem is focused on deducing the joint motions when the position of the end-effector $\boldsymbol{T_e^b}$ is known. If the coordinates of point B_i in the fixed frame are represented by vector $\boldsymbol{b_i}$, then

$$\boldsymbol{p_i} = [x_i, y_i, z_i]^T i = 1, \ \ldots, \ 6 \tag{1}$$

$$r'_{\boldsymbol{i}} = [R_p \cos\theta_{pi}, R_p \sin\theta_{pi}, 0]^T \tag{2}$$

$$\boldsymbol{p} = [x, y, z]^T \tag{3}$$

$$\boldsymbol{b_i} = [R_b \cos\theta_{bi}, R_b \sin\theta_{bi}, 0]^T \tag{4}$$

where $\boldsymbol{p_i}$ is the position vector of point P_i expressed in the fixed coordinate frame, whose coordinates are defined as (x_i,y_i,z_i). $\boldsymbol{r'_i}$ the position vector of point P_i expressed in the moving coordinate frame, and $\boldsymbol{p}$ the position vector of point P expressed in the fixed frame as defined above, and

$$\theta_{\boldsymbol{bi}} = [\theta_{b1}, \theta_{b2}, \theta_{b3}, \theta_{b4}, \theta_{b5}, \theta_{b6}]^T = [T_b, 2\pi/(3 - T_b), /2\pi/(3 + T_b), 4\pi/(3 - T_b), 4\pi/(3 + T_b), -T_b]^T \tag{5}$$

$$\theta_{\boldsymbol{pi}} = [\theta_{p1}, \theta_{p2}, \theta_{p3}, \theta_{p4}, \theta_{p5}, \theta_{p6}]^T = [T_p, 2\pi/(3 - T_p), 2\pi/(3 + T_p), 4\pi/(3 - T_p), 4\pi/(3 + T_p), -T_p]^T \tag{6}$$

From these equations, one can then deduce

$$\boldsymbol{p_i} = \boldsymbol{p} + \boldsymbol{Q}\boldsymbol{r'_i}, i = 1, \ \ldots, \ 6 \tag{7}$$

where $\boldsymbol{Q}$ is the rotation matrix from the fixed reference frame to the moving coordinate frame, and T_p and T_b the angles for determining the attachment points of six symmetrical branched chains on the base and platform, respectively.

Subtracting vector $\boldsymbol{b_i}$ from both sides of Eq. (7), one obtains

$$\boldsymbol{p_i} - \boldsymbol{b_i} = \boldsymbol{p} + \boldsymbol{Q}\boldsymbol{r'_i} - \boldsymbol{b_i} i = 1, \ \ldots, \ 6 \tag{8}$$

Then, taking the Euclidean norm on both sides of Eq. (8), one has

$$\|\boldsymbol{p_i} - \boldsymbol{b_i}\| = \|\boldsymbol{p} + \boldsymbol{Q}\boldsymbol{r'_i} - \boldsymbol{b_i}\| = \boldsymbol{\rho_i} i = 1, \ \ldots, \ 6 \tag{9}$$

where $\boldsymbol{\rho_i}$ is the length of the ith leg or the value of the ith joint coordinate. The solution of the inverse kinematic problem for the six-DOF manipulator is therefore completed and can be written as

$$\boldsymbol{\rho_i^2} = (\boldsymbol{p_i} - \boldsymbol{b_i})^T(\boldsymbol{p_i} - \boldsymbol{b_i}), i = 1, \ \ldots, \ 6 \tag{10}$$

2.3. Jacobian matrix

Based on the parallel component of the mechanism, the parallel Jacobian Matrix can be obtained by differentiating Eq. (10) with respect to time, which obtains

$$\boldsymbol{\rho_i}\dot{\boldsymbol{\rho}}_{\boldsymbol{i}} = (\boldsymbol{p_i} - \boldsymbol{b_i})^T\dot{\boldsymbol{p}}_{\boldsymbol{i}}, i = 1, \ \ldots, \ 6 \tag{11}$$

Since one has

$$\dot{\boldsymbol{Q}} = \boldsymbol{\Omega} \cdot \boldsymbol{Q} \tag{12}$$

with

$$\boldsymbol{\Omega} = 1 \times \boldsymbol{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{13}$$

where $\boldsymbol{\omega}$ is the angular velocity of the platform. Differentiating Eq. (7), one obtains

$$\dot{\boldsymbol{p}}_{\boldsymbol{i}} = \dot{\boldsymbol{p}} + \dot{\boldsymbol{Q}}\boldsymbol{r'_i} \tag{14}$$

Then, Eq. (11) can be rewritten as

$$\begin{aligned} \boldsymbol{\rho_i}\dot{\boldsymbol{\rho}}_{\boldsymbol{i}} &= (\boldsymbol{p_i} - \boldsymbol{b_i})^T(\dot{\boldsymbol{p}} + \dot{\boldsymbol{Q}}\boldsymbol{r'_i}) = (\boldsymbol{p_i} - \boldsymbol{b_i})^T(\dot{\boldsymbol{p}} + \boldsymbol{\Omega}\boldsymbol{Q}\boldsymbol{r'_i}) \\ &= (\boldsymbol{p_i} - \boldsymbol{b_i})^T\dot{\boldsymbol{p}} + (\boldsymbol{p_i} - \boldsymbol{b_i})^T\boldsymbol{\Omega}\boldsymbol{Q}\boldsymbol{r'_i} = (\boldsymbol{p_i} - \boldsymbol{b_i})^T\dot{\boldsymbol{p}} \\ &+ (\boldsymbol{p_i} - \boldsymbol{b_i})^T[\boldsymbol{\omega} \times (\boldsymbol{Q}\boldsymbol{r'_i})] = (\boldsymbol{p_i} - \boldsymbol{b_i})^T\dot{\boldsymbol{p}} + [(\boldsymbol{Q}\boldsymbol{r'}) \\ &\times (\boldsymbol{p_i} - \boldsymbol{b_i})]^T\boldsymbol{\omega}, i = 1, \ \ldots, \ 6 \end{aligned} \tag{15}$$

Hence, one can write the velocity equation as

$$\boldsymbol{A}\boldsymbol{t} = \boldsymbol{B}\dot{\boldsymbol{\rho}} \tag{16}$$

where vector $\boldsymbol{t}$ is the twist of the platform, $\boldsymbol{t} = [\boldsymbol{\omega^T}\dot{\boldsymbol{p}}^{\boldsymbol{T}}]^{\boldsymbol{T}}$, and vector$\dot{\boldsymbol{\rho}}$is defined as

$$\dot{\boldsymbol{\rho}} = [\dot{\boldsymbol{\rho}}_1\dot{\boldsymbol{\rho}}_2 \ \ldots \ \dot{\boldsymbol{\rho}}_6]^T \tag{17}$$

and

$$\boldsymbol{A} = [\boldsymbol{m}_1\boldsymbol{m}_2 \ \ldots \ \boldsymbol{m}_6]^T \tag{18}$$

$$\boldsymbol{B} = diag[\boldsymbol{\rho}_1\boldsymbol{\rho}_2 \ \ldots \ \boldsymbol{\rho}_6]^T \tag{19}$$

where $\boldsymbol{m_i}$ is a vector with 6 components, which can be expressed as

$$\boldsymbol{m_i} = \begin{bmatrix} (\boldsymbol{Q}\boldsymbol{r'}_i) \times (\boldsymbol{p_i} - \boldsymbol{b_i}) \\ (\boldsymbol{p_i} - \boldsymbol{b_i}) \end{bmatrix} \tag{20}$$

The linear transformation between the speed of the manipulator and each joint can be defined as the Jacobian matrix of the robot. This Jacobian matrix indicates the velocity ratio from the space of the joints to the space of the end-effector. According to Eq. (16), the Jacobian matrix can be written as

$$\boldsymbol{J} = \boldsymbol{B}^{-1}\boldsymbol{A} \tag{21}$$

Therefore, the relationship between the Cartesian velocities and joint rates is determined.

3. Design optimization

3.1. Optimization principles

The goal of structural parameters design, also known as dimensional synthesis, is to discover the optimal geometric configuration according to objective functions and geometric restrictions. Specifically, optimization-based dimensional synthesis, one of the significant steps in the design process, ensures that the parallel manipulator will perform strongly in areas such as system stiffness and dexterity.

For parallel manipulators, the aim of optimization is to maximize the system stiffness and minimize the dexterity, which

is expressed with the condition number in the Jacobian matrix. If these two aspects are addressed separately, then it will be treated as two SOO issues. Contrarily, if the two aspects are considered together, then it will be considered a MOO issue.

Due to the lack of convergence, only a few geometrical parameters can be managed at one time. The lack of convergence exists because traditional optimization methods use a local search by a convergent stepwise procedure, such as gradient, Hessians, linearity, and continuity, which compare the values of the subsequent points and then examine the relative optimal points [25]. Global optima can only be found if the problem possesses certain convexity properties, which essentially guarantee that any local optima are a global optimum. In other words, conventional methods are based on a point-to-point rule, which has the danger of failing in the case of local optima.

On the other hand, genetic algorithms are based on the population-to-population rule, which can avoid the problems caused by local optima. Genetic algorithms have the advantages of robustness and strong convergence properties, which include the following:

- They require no knowledge or gradient information about the optimization problems; only the objective function and corresponding fitness levels influence the directions of the search.
- Discontinuities present in the optimization problems have little effect on the overall optimization performance.
- They are generally more straightforward to introduce, since there are no restrictions for the definition of the objective function.
- They use probabilistic transition rules rather than deterministic ones.
- They perform well in large-scale optimization problems.

Genetic algorithms have been shown to solve linear and nonlinear problems by exploring all regions of state space and exponentially exploiting promising areas through mutation, crossover, and selection operations that are applied to individuals in the population. Since evolution is an inherently parallel process, the greatest potential for the application of evolutionary optimization to real-world problems will entail their implementation on parallel machines [26,27].

Although a single population genetic algorithm performs well on a wide variety of problems, for multi-variable and complicated optimization problems, it is inconvenient to find optima using limited population size and evolutionary generations [28]. Alternatively, more effective results can be obtained by introducing multiple sub-populations. For instance, Fig. 4 depicts the behavioral rationale for the extended multi-population genetic algorithm, which is adopted in this research.

One problem with implementing genetic algorithms concerns how to model the objective function. Although genetic algorithms can be used to search the best solution set, it is very difficult and time-consuming, especially when the parameters are diverse and the objective functions are too complex for genetic algorithms to work effectively based on the analytical expression of the performance indices, especially in the case of MOO. Consequently, neural networks will be applied to address this problem. In particular, the output error for neural networks is constrained in a minimal threshold value that will not affect the computing accuracy with CPU.

Artificial neural networks are parallel adaptive networks consisting of simple nonlinear computing elements called neurons. Neurons are intended to abstract and model some functions of the human nervous system to simulate its powerful computation ability. A network is viewed as a weighted directed graph, where artificial neurons are the basic elements and directed weighted edges represent connections between neurons. Networks are used to calculate the training error for the cost function, which is done by performing a random initialization of weights and by training the network with one of the most commonly used learning algorithms, such as backpropagation [29,30]. The basic element of neural networks is illustrated in Fig. 5.

The following equation describes the relationship among inputs and outputs for the artificial neuron:

$$y_i = f_i\left(\sum_{t=1}^{n} x_t w_t + b_i\right) - \theta_i \tag{22}$$

where f_i is the transfer function, such as the sigmoid function, x_t the input signal, which is the corresponding output from the anterior neuron, w_t the weight value of the related input signal, b_i the bias, whose weight value is 1, and θ_i the threshold value. The overall structure of neural networks is organized with many neurons arranged in a specific order. Subsequently, the corresponding learning rule should be confirmed for the training process.

The main function of neural networks is to establish a complex nonlinear relationship between the inputs and outputs without deducing a mathematical expression, which is referred to as a

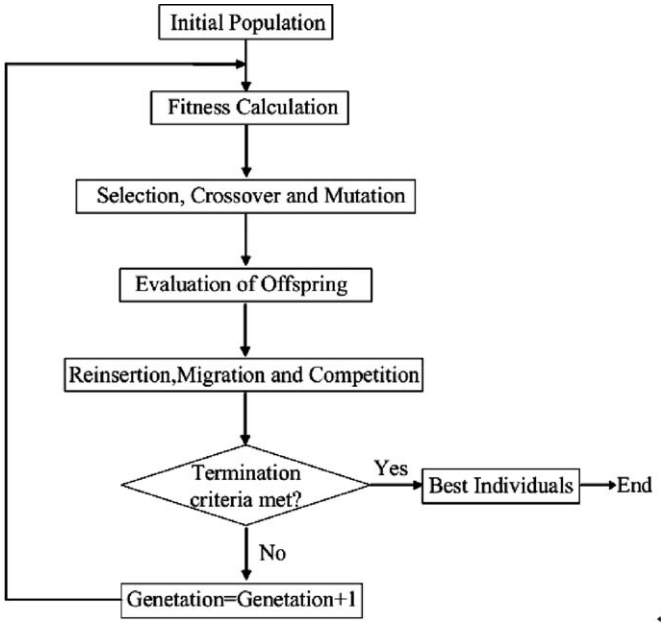

Fig. 4. Schematic representation of the optimization rationale based on genetic algorithms.

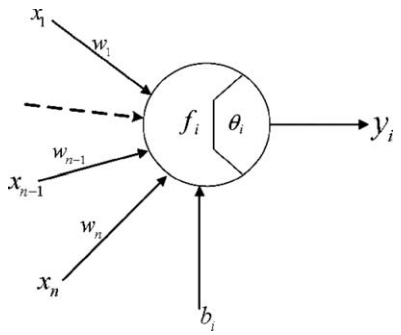

Fig. 5. The basic element of neural networks.

"black box". For real-world applications, the primary properties of neural networks include the capacities for associative recall and function approximation.

Furthermore, neural networks are widely noted for their model-free estimation capability, as they are able to create internal representations solely through training example sets and without the use of a mathematical model depicting the relationship between inputs and outputs. Sometimes, this ability is referred to as similarity based generalization. Using powerful learning algorithms, neural networks are able to approximate functions by sifting through vast repositories of data. Because of their learning capability, they are referred to as adaptive function estimators. Therefore, they can be utilized to represent the expressions of system stiffness and dexterity for the six-DOF parallel manipulator.

3.2. Solution for system stiffness

From the viewpoint of mechanics, stiffness is the measured ability of a body or structure to resist deformation due to the action of external forces. Specifically, the stiffness of a parallel mechanism at a given point in its workspace can be characterized by its stiffness matrix. This matrix relates the forces and torques applied at the gripper link in Cartesian space to the corresponding linear and angular Cartesian displacements.

Two primary methods have been used to establish mechanism stiffness models: first, matrix structural analysis models structures as a combination of elements and nodes. Alternatively, the second method relies on calculating the Jacobian matrix for the parallel mechanism; accordingly, this latter method is adopted in this research.

For the parallel mechanisms in this work, the velocity relationship can be written as in Eq. (23)

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}\dot{\boldsymbol{x}} \tag{23}$$

where $\dot{\boldsymbol{\theta}}$ is the vector of joint rates and $\dot{\boldsymbol{x}}$ is the vector of Cartesian rates. This latter variable is a six-dimensional twist vector containing the velocity and angular velocity of a point on the platform. Matrix $\boldsymbol{J}$ is usually termed the Jacobian matrix, which is described in Eq. (21).

The stiffness matrix of the mechanism in Cartesian space is then given by the following expression:

$$\boldsymbol{K_C} = \boldsymbol{J^T K_J J} \tag{24}$$

In this case, $\boldsymbol{K_J}$ is the joint stiffness matrix of the parallel mechanism. Specifically, $\boldsymbol{K_J} = diag[k_1, \ldots, k_n]$, where each of the actuators in the parallel mechanism is modeled as an elastic component. Furthermore, k_i is a scalar representing the joint stiffness of each actuator, which is modeled as a linear spring.

In the case where all actuators have the same stiffness, such as, $k = k_1 = k_2 = \cdots = k_n$, then Eq. (24) will be reduced to

$$\boldsymbol{K_C} = k\boldsymbol{J^T J} \tag{25}$$

Moreover, the diagonal elements of the stiffness matrix are used as the system stiffness value. These elements represent the pure stiffness in each direction as well as reflecting the rigidity of machine tools more clearly and concisely. The objective function for system stiffness optimization can be written as

$$Stiffness\ \ Val = \sum_{i=1}^{6} \eta_i K_{ii} \tag{26}$$

where for $i = 1, \ldots, 6$, K_{ii} represents the diagonal elements of the mechanism's stiffness matrix, η_I the weight factor for each directional stiffness, which characterizes the priority of the stiffness in this direction. For the optimization of system stiffness, this factor should be a maximum.

3.3. Solution for dexterity

The condition number of the Jacobian matrix will be a measure of the dexterity indices for the six-DOF parallel manipulator

$$Dexterity = Cond(\boldsymbol{J}) \tag{27}$$

where $Cond(\boldsymbol{J})$ is the condition number of the Jacobian matrix and is defined as

$$Cond(\boldsymbol{J}) = \|\boldsymbol{J}\|\|\boldsymbol{J}^{-1}\| \tag{28}$$

where $\|\cdot\|$ represents the norm of the related vector or matrix. If the Frobenius norm is considered, then

$$\|\boldsymbol{J}\| = \sqrt{tr(\boldsymbol{J^T J})} \tag{29}$$

In this case, the Frobenius norm is defined as the extracting roots of the quadratic sum for each element in the Jacobian matrix. Hence, the dexterity indices can be deduced as

$$Cond(\boldsymbol{J}) = \sqrt{tr(\boldsymbol{J^T J})} \cdot \sqrt{tr((\boldsymbol{J}^{-1})^{\boldsymbol{T}} \cdot \boldsymbol{J}^{-1})} \tag{30}$$

If the spectral norm is introduced, the dexterity indices will be described as

$$Cond(\boldsymbol{J}) = \frac{SV_{\max}(\boldsymbol{J})}{SV_{\min}(\boldsymbol{J})} \tag{31}$$

where $SV_{\max}(\boldsymbol{J})$ and $SV_{\min}(\boldsymbol{J})$ represent the singular maximum and minimum values of Jacobian matrix $\boldsymbol{J}$, respectively. This expression is selected as the objective function for the optimization of dexterity, especially because of its efficient computing time. The value of $Cond(\boldsymbol{J})$, which is directly related to the singular values of the Jacobian matrix, is between one and positive infinity. All singular values in the Jacobian matrix will be the same and the manipulator is isotropic when $Cond(\boldsymbol{J})$ is equal to 1. On the other hand, when $Cond(\boldsymbol{J})$ is potentially equal to positive infinity, then the Jacobian matrix is singular. Therefore, when the condition number is considered, $Cond(\boldsymbol{J})$ should be a minimum for the optimization of dexterity.

For SOO issues, these two objective functions should be considered, respectively. On the other hand, for a MOO issue, the two goals might be in conflict, so a strategy for addressing both objectives equally should be considered in the optimization process. The integration of neural networks, genetic algorithms, and the Pareto approach can be viewed as one type of evolutionary neural network that searches the optimal solution sets of MOO.

3.4. Multi-objective genetic algorithms

Three main multi-objective genetic algorithms will be introduced as follows:

3.4.1. Clustering function approach

The main principle of this approach involves converting the MOO problem into SOO by distributing the weighting factors of different objective function values.

For instance, if there are objective functions, such as $f_1(x), f_2(x), \ldots, f_r(x)$, it becomes

$$\min \sum_{i=1}^{r} w_i \cdot f_i(x) (i = 1, 2, \ \ldots \ r) \tag{32}$$

where

$$\sum_{i=1}^{r} w_i = 1$$

Then, the multi-objective issue becomes a single-objective problem. A typical application of the clustering function approach can be found in [31].

3.4.2. Population-based approach

The population-based approach originates from the conception of population division [32], where co-evolution, including competition and cooperation, is the basic feature of this approach. However, there are still no effective methods for population decomposition; sub-population size determination, representative selection, and their application modes and domains require further expansion. The act of decomposing the co-evolutionary population is based on using piecewise interval correlation with the iteration linkage learning method. For multi-population genetic algorithms, the strategy for dynamically changing the search area depends on the distribution of the best individual in each population. The adaptive adjustment method of the population size is based on the dimensions of the search area. For instance, a typical application of the population-based approach can be found in [33].

3.4.3. Pareto-based approach

The concept of the Pareto Method was originally introduced by Francis Ysidro, and subsequently, it was generalized by Vilfredo Pareto [34]. Basically, the Pareto set includes the best solutions when there are no other results that can improve at least one of the objectives without degrading any other objective. Usually, there will be a set of solutions that provides a maximum amount of information about the optimization for multi-objective problems. After comparing each solution to every other one, those solutions that satisfy the least number of objectives are flagged as inferior [35–37]. For maximizing the k objective functions, the decision vector (sets of variables) $\boldsymbol{x}^* \in F$ is the Pareto-optimal solution if no other decision vectors satisfy both of the following conditions:

$$f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}^*), \forall i \in \{1, 2, \ldots, k\} \tag{33}$$

$$f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}^*), \exists j \in \{1, 2, \ldots, k\} \tag{34}$$

Likewise, if both of the following conditions are true, decision vector $\boldsymbol{x}$ dominates $\boldsymbol{y}$ in the maximization issue, as noted by $\boldsymbol{x} > \boldsymbol{y}$. This is expressed as

$$f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{y}), \forall i \in \{1, 2, \ldots, k\} \tag{35}$$

$$f_j(\boldsymbol{x}) > f_j(\boldsymbol{x}^*), \exists j \in \{1, 2, \ldots, k\} \tag{36}$$

According to Eqs. (33)–(36), the Pareto-optimal set can be defined in the following manner: if there is no solution in the search space that dominates any member in the set $\boldsymbol{P}$, then the solutions belonging to the set $\boldsymbol{P}$ constitute a global Pareto-optimal set. An application of the Pareto-based approach, which is the method adopted in this research, can be found in [38].

4. Simulation

4.1. Objective optimization of system stiffness

Five architectural and behavioral parameters are used as the optimization parameters for obtaining the maximum system stiffness of the spatial six-DOF parallel manipulator, as shown in Figs. 1 and 2. The vector of optimization variables is expressed as

$$\boldsymbol{s} = \{R_P, R_b, z, T_p, T_b\} \tag{37}$$

where R_p, R_b are the radius of the moving platform and the base, respectively, z the height of the platform, T_p and T_b the angles for determining the attachment points on the base and on the platform, and their boundaries are shown in Table 1.

As the most popular training method for the feedforward neural network, the standard backpropagation learning algorithm is based on the steepest descent gradient approach to the minimization of a criterion function representing the instantaneous error between the target output and the predicted output. The criterion function can be expressed as follows:

$$E = \frac{1}{2N}(\boldsymbol{T_{out}} - \boldsymbol{Y_{out}})^T(\boldsymbol{T_{out}} - \boldsymbol{Y_{out}}) \tag{38}$$

where $\boldsymbol{T_{out}}$ is the vector of the desired network output, $\boldsymbol{Y_{out}}$ the vector of the actual output, and N the vector dimension. In this scenario, the five geometrical parameters of this six-DOF parallel manipulator are chosen as the inputs of the feedforward neural network, and the performance index is used as the output.

The basic training step of a neural network with the standard backpropagation algorithm is:

(A) Initialize the weights and biases in each layer with small random values to ensure that the weighted inputs of the network will not be saturated.
(B) Confirm the set of input/output pairs and the network structure. Set some of the related parameters, such as the desired minimal E, the maximum number of iterations and the learning speed.
(C) Compare the actual output with the desired network response and calculate the deviation.
(D) Train the updated weights based on the criterion function in each epoch.
(E) Continue the previous two steps until the network satisfies the training requirement.

Fig. 6 depicts the topology of the neural network, which is developed as the objective function for modeling the analytical solution of system stiffness. In this case, two hidden layers with the sigmoid transfer function are established, where 16 neurons exist in each hidden layer. The input vectors include the random arrangement of discretization values from the five structural variables.

Fig. 7 illustrates the training result from using the standard backpropagation learning algorithm, where the blue curve denotes the quadratic sum of the output errors with respect to the ideal output values. After the neural network has trained for 206 times, it reaches the target error goal.

Table 1
Variables of structural parameters.

R_p	(0.05, 0.1) m
R_b	(0.12, 0.22) m
Z	(0.16, 0.26) m
T_p	(18°, 28°)
T_b	(38°, 48°)

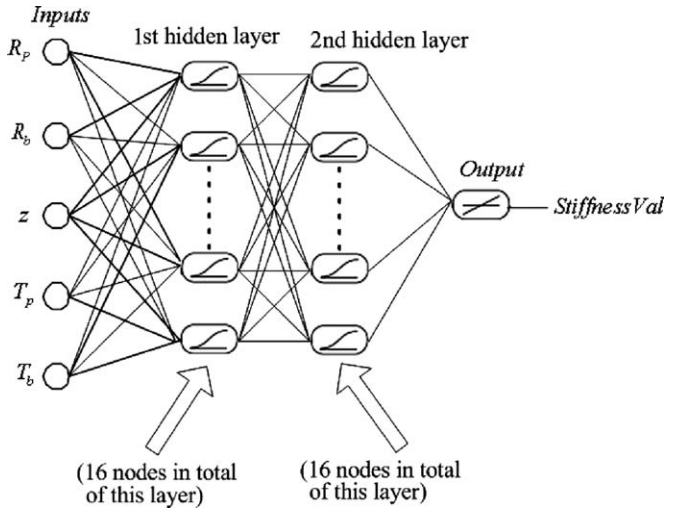

Fig. 6. The topology of the feedforward neural network for the solution of system stiffness.

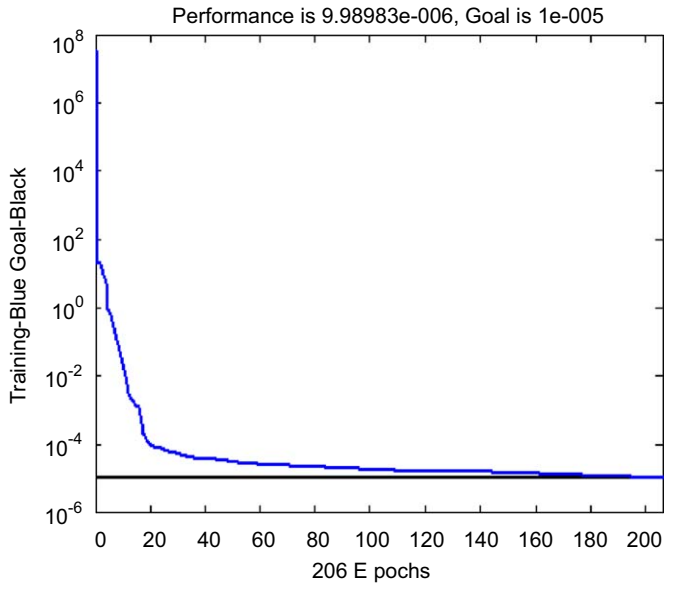

Fig. 7. Training result of the objective function for system stiffness using the BP neural network.

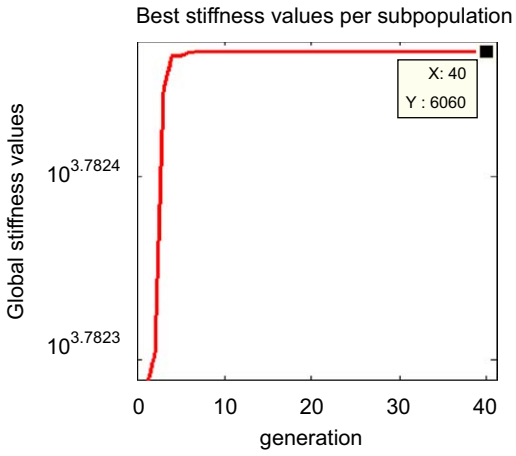

Fig. 8. The evolution of system stiffness.

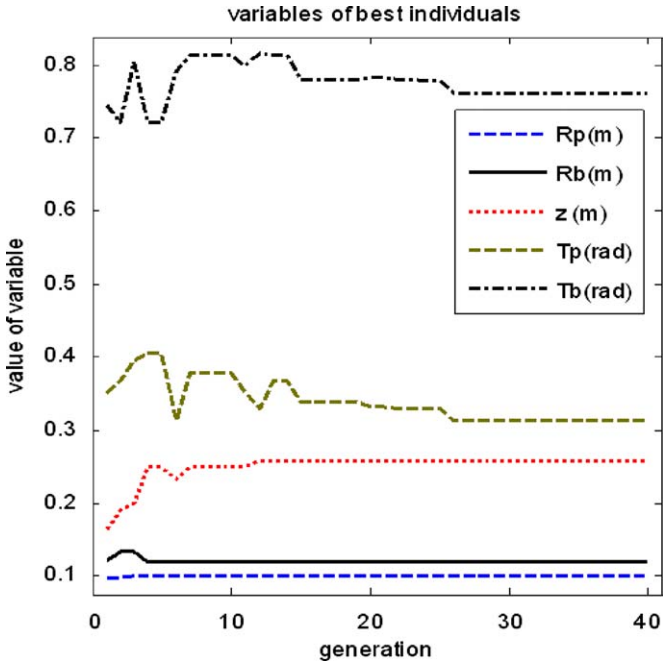

Fig. 9. The evolution of architectural and behavioral parameters for system stiffness optimization.

After the trained neural network is ready for the objective function, the genetic algorithm can be implemented to search for the best solutions. The evolution of the best individual over 40 generations is depicted in Fig. 8, where the optimal system stiffness value is 6060.

In terms of architectural and behavioral variables, the evolution for the best individuals in the population is shown in Fig. 9. By simultaneously adjusting the five parameters, the optimization results for system stiffness are obtained. After 40 generations, they are convergent as follows:

$$\boldsymbol{s} = \{R_P, R_b, z, T_p, T_b\} = \{0.1\,\text{m}, 0.12\,\text{m}, 0.26\,\text{m}, 0.31416\,\text{rad}, 0.74837\,\text{rad}\}$$

4.2. Objective optimization of dexterity

In some respects, the optimization of dexterity differs from that of system stiffness. Since the analytical expression of the dexterity indices is more complex than the case of system stiffness, computing time and error accuracy should be prioritized for the design of the neural network's structure and the choice of its learning algorithm.

As shown in Fig. 10, a neural network containing a single hidden layer of 100 neurons is developed. Using more neurons and fewer hidden layers assists in reducing the computing time and improving the error accuracy. Additionally, this study introduces an improved training method, the Levenberg–Marquardt (LM) algorithm, which disregards directions in the parameter spaces that marginally influence the criterion and increases the learning rate. Besides, it still performs similarly to the efficient Gauss–Newton directions within the subset of the important parameters [39]. The core of the LM algorithm is the calculus and inversion of an approximation to the Hessian in each training cycle [40]. This method improves the solution to problems that are much more difficult to solve by repeatedly adjusting the learning rate. It can be expressed as

$$f(\boldsymbol{X}^{(\boldsymbol{k}+1)}) = \min f(\boldsymbol{X}^{(\boldsymbol{k})} + \eta^{(k)} S(\boldsymbol{X}^{(\boldsymbol{k})})) \tag{39}$$

$$\boldsymbol{X}^{(\boldsymbol{k}+1)} = \boldsymbol{X}^{(\boldsymbol{k})} + \eta^{(k)} S(\boldsymbol{X}^{(\boldsymbol{k})}) \tag{40}$$

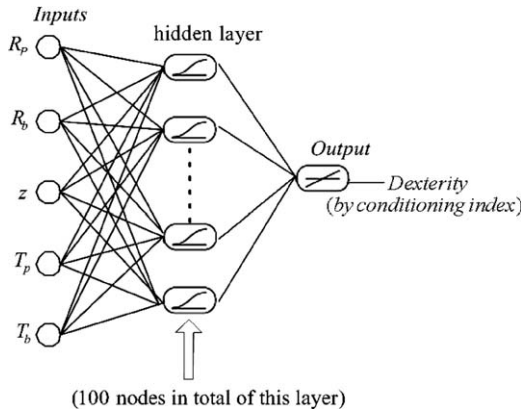

Fig. 10. The topology of the feedforward neural network for the solution of dexterity.

where $\boldsymbol{X}^{(k)}$ is the vector constituting all of the weights and biases of the neural network, $S(\boldsymbol{X}^{(k)})$ the search direction for each component of $\boldsymbol{X}$, $\eta^{(k)}$ the step index for minimizing $f(\boldsymbol{X}^{(k+1)})$ in the direction of $S(\boldsymbol{X}^{(k)})$. The Levenberg–Marquardt algorithm integrates the steepest descent gradient approach with the Newton method, which is a common numerical optimization method. Its searching direction is represented by

$$S(\boldsymbol{X}^{(k)}) = -(\boldsymbol{H}^{(k)} + \lambda^{(k)}\boldsymbol{I})^{-1}\nabla f(\boldsymbol{X}^{(k)}) \tag{41}$$

where $\boldsymbol{H}$ is the Hessian matrix, $\boldsymbol{I}$ an identity matrix, and λ produces a conditioning effect, which is automatically selected until a downhill step is produced for each stage. Therefore, this modified algorithm of backpropagation is adopted for the optimization of dexterity.

Fig. 11 describes the training result using a feedforward neural network with the Levenberg–Marquardt learning algorithm. In this figure, the blue curve denotes the quadratic sum of the output errors with respect to the ideal values. Specifically, the target error of 10^{-8} is much smaller than that of the training result for system stiffness, which is 10^{-5}. Furthermore, the neural network achieves its target error after training 82 times. Therefore, by using the Levenberg–Marquardt algorithm to train the neural network, the number of repetitions is decreased while the computational precision is vastly improved.

The optimization process for dexterity based on the condition number of the Jacobian matrix is illustrated in Fig. 12. After a global search of 40 generations, the optimal dexterity value converges at 353.1.

The evolution of architectural and behavioral parameters for dexterity optimization is described in Fig. 13. In comparing this figure to Fig. 9, it is evident that the corresponding convergent points for the five parameters in these two figures are different.

$$s = \{R_P, R_b, z, T_p, T_b\}$$
$$= \{0.1\,\text{m}, 0.22\,\text{m}, 0.16\,\text{m}, 0.31416\,\text{rad}, 0.83733\,\text{rad}\}$$

More precisely, it is evident that the two objective functions conflict with each other. This issue will be discussed in the following section, where the Pareto-optimal solution for MOO is addressed.

4.3. Multi-objective optimization

MOO problems consist of simultaneously optimizing several objective functions that differ from those of SOO. For a SOO task, one single global optimal search is adequate. However, in a MOO problem, it is necessary to find all possible compromises among

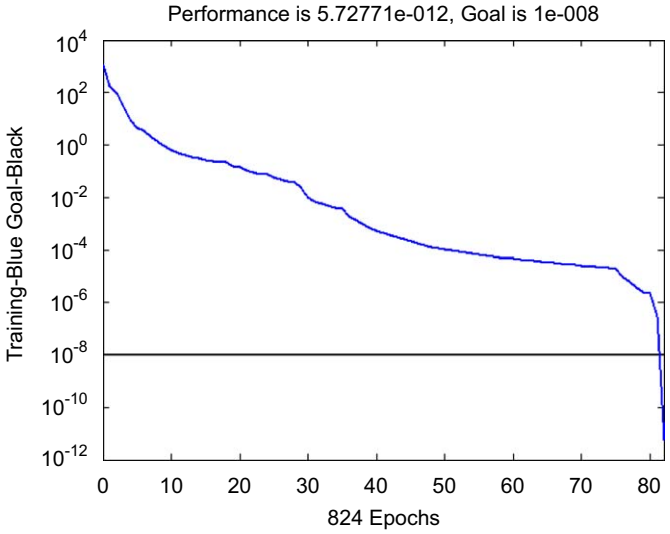

Fig. 11. Dexterity training result for the objective function by using the feedforward neural network with the improved learning algorithm.

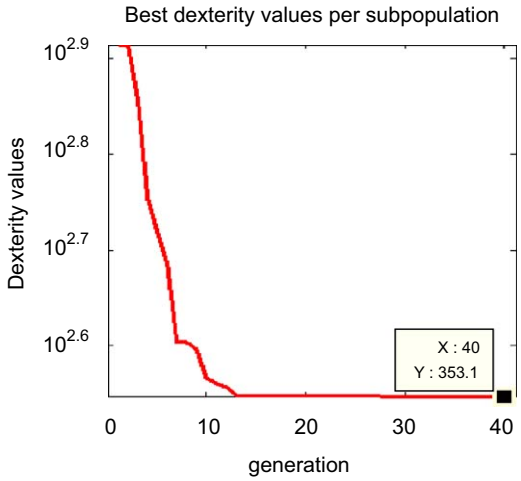

Fig. 12. The evolution of dexterity.

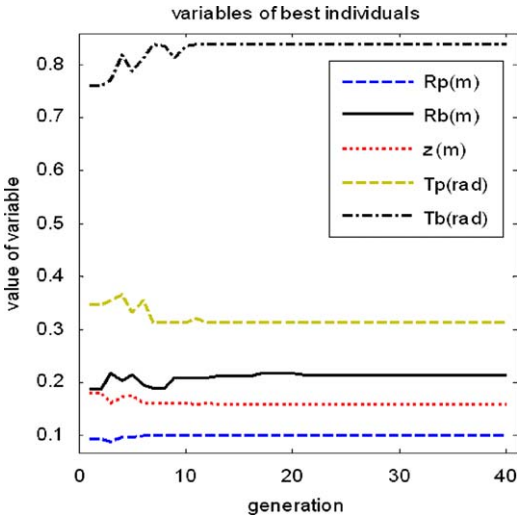

Fig. 13. The evolution of architectural and behavioral parameters for dexterity optimization.

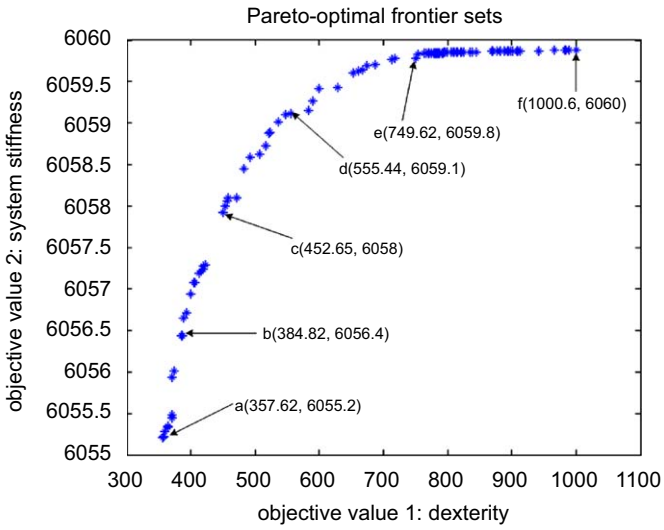

Fig. 14. Pareto-optimal solutions and the pareto frontier in the solution space.

Table 2
Objective functions values from pareto sets and the corresponding design variables.

	Dexterity/system stiffness	R_p (m)	R_b (m)	zZ (m)	T_p (rad)	T_b (rad)
a	357.62/6055.2	0.1	0.20928	0.16064	0.31416	0.83762
b	384.82/6056.4	0.1	0.19281	0.16	0.31416	0.83448
c	452.65/6058	0.1	0.17055	0.16021	0.31442	0.83775
d	555.44/6059.1	0.1	0.1498	0.16149	0.31416	0.83775
e	749.62/6059.8	0.1	0.12739	0.16426	0.31483	0.8377
f	1000.6/6060	0.1	0.12006	0.17632	0.31956	0.82563

multiple objective functions that conflict with each other. Consequently, the set of Pareto-optimal solutions is generally used to make the final decision.

Prior to implementation, the following initial parameters of the Pareto-based genetic algorithms are created:

Number of sub-populations 5
Number of individuals in each sub-population {50, 30, 30, 40, 50}
Mutation range 0.01
Mutation precision 24
Maximum generations for algorithm termination 300

After optimization, all possible solutions in the entire solution space are obtained without the need for combining all objective functions into one. Fig. 14 shows the Pareto-optimal frontier sets, where the designers can determine the final solutions depending on their preferences. Hence, the analysis process and cycle time is greatly reduced. In this figure, a compromise between the objectives of system stiffness and dexterity is demonstrated in the distributing trend of these Pareto points. If any other pair of design variables is chosen from the lower-right area in Fig. 14, its corresponding values will locate an inferior point with respect to the Pareto frontier. Furthermore, the upper-left side is an inaccessible area for all the possible solution pairs, which is the reason why Pareto solutions are called Pareto-optimal frontier sets. Lastly, there are six optimum design points, labeled "a" to "f," whose corresponding objective values and design parameters are shown in Table 2.

5. Conclusions

Performance optimization is an integral issue for the more extensive industrial applications of parallel manipulators. By adjusting the architectural and behavioral parameters, the optimal performance indices of the six-DOF parallel mechanism, such as system stiffness and dexterity, can be searched. Only a few geometrical parameters can be managed due to the lack of convergence of the optimization algorithm for complex objective functions with traditional optimization methods. Consequently, neural networks and genetic algorithms are the two important artificial intelligence approaches simultaneously implemented as the optimization guidelines in this paper. Single-objective and multi-objective optimization issues are both addressed to demonstrate the validity of the artificial intelligence approaches, both of which are naturally compatible. Specifically, in the case of MOO, Pareto evolutionary neural networks possess high efficiency, strong generalization, and programmable features regardless of whether the objective functions have an analytical solution. Future work will focus on the comprehensive MOO problem for complicated parallel mechanisms with the objective of realizing the simultaneous optimization of additional performance indices, such as system stiffness, workspace, dexterity, and malleability. In particular, the physical parallel manipulator is under development using the proposed performance evaluation criteria.

Acknowledgments

The authors would like to acknowledge the financial support from the National Science and Engineering Research Council of Canada (NSERC). The second author gratefully acknowledges the financial support from the Canada Research Chairs program.

References

[1] Steward D. A platform with six degrees of freedom. Proc Inst Mech Eng 1965;180(5):371–86.
[2] Hostens I, Anthonis J, Ramon H. New design for a 6 DOF vibration simulator with improved reliability and performance. Mech Syst Signal Process 2005;19(1):105–22.
[3] Carretero JA, Podhorodeski RP, Nahon MN, Gosselin CM. Kinematic analysis and optimization of a new three degree-of-freedom spatial parallel manipulator. ASME J Mech Des 2000;122(1):17–24.
[4] Dunlop GR, Jones TP. Position analysis of a two DOF parallel mechanism—Canterbury tracker. Mech Mach Theory 1999;34(1):599–14.
[5] Lee KM, Arjunan S. A three-degrees-of-freedom micromotion in-parallel actuated manipulator. IEEE Trans Rob Autom 1991;7(5):634–41.
[6] Jensen KA, Lusk CP, Howell LL. An XYZ micromanipulator with three translational degrees of freedom. Robotica 2006;24(3):305–14.
[7] Xu QS, Li YM. A novel design of a 3-PRC compliant parallel micromanipulator for nanomanipulation. Robotica 2006;24(4):527–8.
[8] Fedewa D, Mehrabi MG, Kota S, Orlandea N, Gopalakrishnan V. Parallel structures and their applications in reconfigurable machining systems. In: Proceedings of parallel kinematic machines—international conference, 2000. pp. 87–97.
[9] Nguyen CC, Zhou ZL, Bryfogis M. A robotically assisted munition loading system. J Rob Syst 1995;12(12):871–81.
[10] Yang GL, Chen IM, Chen WH, Lin W. Kinematic design of a six-DOF parallel-kinematics machine with decoupled-motion architecture. IEEE Trans Rob Autom 2004;20(5):876–84.
[11] Li YM, Xu QS. Stiffness analysis for a 3-PUU parallel kinematic machine. Mech Mach Theory 2008;43(2):186–200.
[12] Chen SL, Chang TH, Lin YC. Applications of equivalent components concept on the singularity analysis of TRR-XY hybrid parallel kinematic machine tools. Int J Adv Manuf Technol 2006;30(7–8):778–88.
[13] Refaat S, Herve JM, Nahavandi S. Two-mode overconstrained three-DOFs rotational-translational linear-motor-based parallel-kinematics mechanism for machine tool applications. Robotica 2007;25(4):461–6.
[14] Zhang D, Xi F, Mechefske C, Sherman YT. Analysis of parallel kinematic machines with kinetostatic modelling method. Rob Comput-Integr Manuf 2004;20(2):151–65.
[15] Staicu S, Zhang D. Dynamic modeling of a 4-DOF parallel kinematic machine with revolute actuators. Int J Manuf Res 2008;3(2):172–87.
[16] Zhang D, Wang L, Esmailzadeh E. PKM capabilities and applications exploration in a collaborative virtual environment. Rob Comput-Integr Manuf 2006;22(4):384–95.
[17] Bergamaschi PR, Nogueira AC, Saramago FP. Design and optimization of 3R manipulators using the workspace features. Appl Math Comput 2006; 172(1):439–63.

[18] Rout BK, Mittal RK. Parametric design optimization of 2-DOF R–R planar manipulator: a design of experiment approach. Rob Comput-Integr Manuf 2008;24(2):239–48.
[19] Yu A, Bonev IA, Paul ZM. Geometric approach to the accuracy analysis of a class of 3-DOF planar parallel robots. Mech Mach Theory 2008;43(3): 364–375.
[20] Zhao JS, Zhang SL, Dong JX, Feng ZJ, Zhou K. Optimizing the kinematic chains for a spatial parallel manipulator via searching the desired dexterous workspace. Rob Comput-Integr Manuf 2007;23(1):38–46.
[21] Stock M, Miller K. Optimal kinematic design of spatial parallel manipulators: application to linear delta robot. ASME J Mech Des 2003;125: 292–301.
[22] Kucuk S, Bingul Z. Robot workspace optimization based on a novel local and global performance indices. In: IEEE International Symposium on Industrial Electronics, 2005. pp. 1593–8.
[23] Ceccarelli M, Lanni C. A multi-objective optimum design of general 3R manipulators for prescribed workspace limits. Mech Mach Theory 2004;39(2):119–32.
[24] Holland JH. Adaptation in natural and artificial systems. Ann Arbor, MI: The University of Michigan Press; 1975.
[25] Gosselin CM, Guillot M. The synthesis of manipulators with prescribed workspace. ASME J Mech Des 1991;113(1):451–5.
[26] Fogel D B. An introduction to simulated evolutionary optimization. IEEE Trans Neural Networks 1994;5(1):3–14.
[27] Mininno E, Cupertino F, Naso D. Real-valued compact genetic algorithms for embedded microcontroller optimization. IEEE Trans Evol Comput 2008;12(2):203–19.
[28] Gong DW, Hao GS, Zhou Y, Sun XY. Interactive genetic algorithms with multi-population adaptive hierarchy and their application in fashion design. Appl Math Comput 2007;185(2):1098–108.
[29] Liu YY, Starzyk JA, Zhu Z. Optimized approximation algorithm in neural networks without overfitting. IEEE Trans Neural Networks 2008;19(6): 983–995.
[30] Ludermir TB, Yamazaki A, Zanchettin C. An optimization methodology for neural network weights and architectures. IEEE Trans Neural Networks 2006;17(6):1452–9.
[31] Zhang D, Wang L, Lang S. Parallel kinematic machines: design, analysis and simulation in an integrated virtual environment. ASME J Mech Des 2005;127(7):580–8.
[32] Potter MA, De Jong KA. Cooperative coevolutionary architecture for evolving coadapted subcomponents. Evol Comput 2000;8(1):1–29.
[33] Pedrajas NG, Martınez CH, Perez JM. Multi-objective cooperative coevolution of artificial neural networks. Neural Networks 2002;15(1):1259–78.
[34] Coello CA, Veldhuizen DA, Lamont GB. Evolutionary algorithms for solving multi-objective problems. New York: Kluwer Academic Publishers; 2002.
[35] Baumgartner U, Magele Ch, Renhart W. Pareto optimality and particle swarm optimization. IEEE Trans Magn 2004;40(2):1172–5.
[36] Papila M, Haftka RT, Nishida T, Sheplak M. Piezoresistive microphone design pareto optimization: tradeoff between sensitivity and noise floor. J Microelectromech Syst 2006;15(6):1632–43.
[37] Fieldsend JE, Singh S. Pareto evolutionary neural networks. IEEE Trans Neural Networks 2005;16(2):338–54.
[38] Gupta S, Tiwari R, Nair SB. Multi-objective design optimization of rolling bearings using genetic algorithms. Mech Mach Theory 2007;42(10):1418–1443.
[39] Ngia LSH, Sjoberg J. Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg–Marquardt algorithm. IEEE Trans Signal Process 2000;48(7):1915–27.
[40] Toledo A, Pinzolas M, Ibarrola JJ, Lera G. Improvement of the neighborhood based Levenberg–Marquardt algorithm by local adaptation of the learning coefficient. IEEE Trans Evol Comput 2005;16(4):988–92.